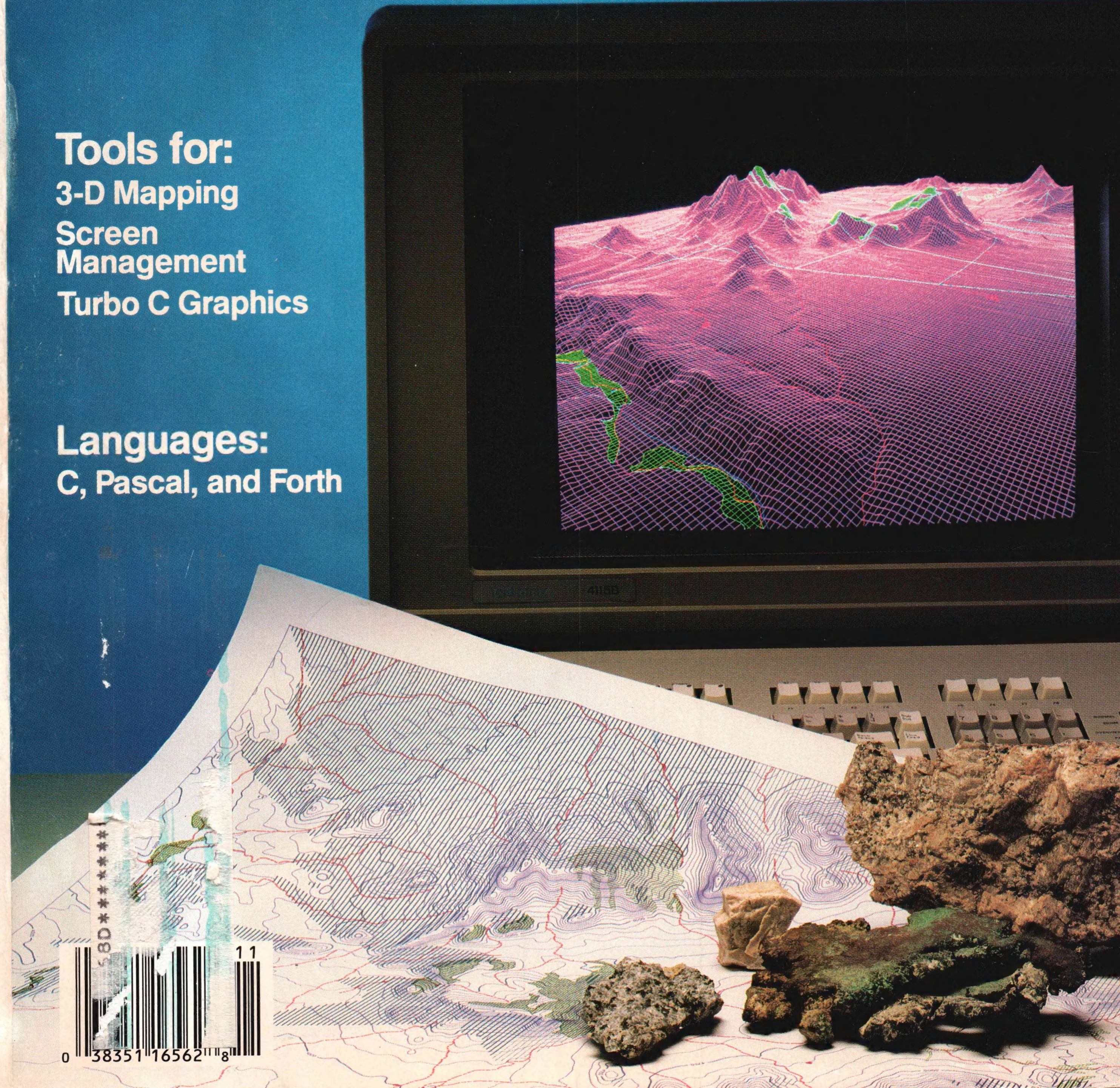
#133 NOVEMBER 1987

2.95 (3.95 CANADA)

Dr. Dobb's Journal of Software Tools

FOR THE PROFESSIONAL PROGRAMMER

Special Graphics Issue



Tools for:
3-D Mapping
Screen Management
Turbo C Graphics

Languages:
C, Pascal, and Forth

er than ever before!

4.0 uses logical units for separate compilation

Pascal 4.0 lets you break up the code gang into "units," or "chunks." These logical modules can be worked with swiftly and separately—so that an error in one module is seeable and fixable, and you're not sent through all your code to find one error. Compiling and linking these separate units happens in a flash because your compiling horsepower is better than 27,000 lines a minute.* And 4.0 also includes an automatic project Make.

4.0's cursor automatically lands on any trouble spot

4.0's interactive error detection and location means that the cursor automatically lands where the error is. While you're compiling or running a program, you get an error message at the top of your screen *and* the cursor flags the error's location for you.

4.0 gives you an integrated programming environment

4.0's integrated environment includes pull-down menus and a built-in editor. Your program output is automatically saved and shown in the output window. You can Scroll, Pan, or Page through all your output and know where everything is all the time. Given 4.0's integration, you can edit, compile, find and correct errors—all from inside the integrated development environment.

You'll never lose your mind, because 4.0 never loses your place

Whenever you re-load 4.0, it remembers what you and it were doing before you left. It puts you right back in the editor with the same file and in the same place as you were working last.

*Run on an 8 MHz IBM AT.

**If within 60 days of purchase this product does not perform in accordance with our claims, call our customer service department, and we will arrange a refund.

Please check box(es)	*Sugg. Retail*	*Upgrade Price†*	Serial No.
☐ Turbo Pascal 4.0 Compiler	$ 99.95	$ 39.95	________
☐ Turbo Pascal 4.0 Developer's Library	395.00	150.00	
(Includes Turbo Pascal Tutor and all Toolboxes; must be ordered with Compiler)			
☐ Turbo Pascal Tutor	69.95	19.95	________
☐ Turbo Pascal Database Toolbox	99.95	29.95	________
☐ Turbo Pascal Graphix Toolbox	99.95	29.95	________
☐ Turbo Pascal Editor Toolbox	99.95	29.95	________
☐ Turbo Pascal Numerical Methods Toolbox	99.95	29.95	________
☐ Turbo Pascal Gameworks	99.95	29.95	________

Total product amount $ ____________
CA and MA residents add sales tax $ ____________
Shipping and handling* $ ____________
Total amount enclosed $ ____________

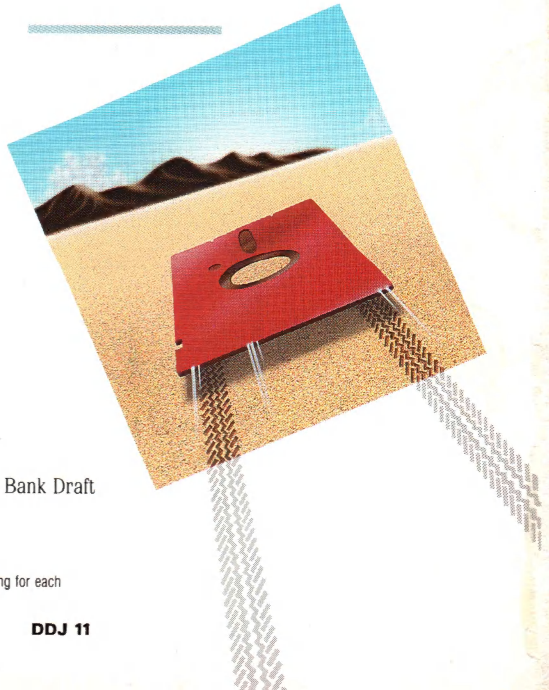
Please specify diskette size: ☐ 5¼″ ☐ 3½″ Payment: ☐ VISA ☐ MC ☐ Check ☐ Bank Draft

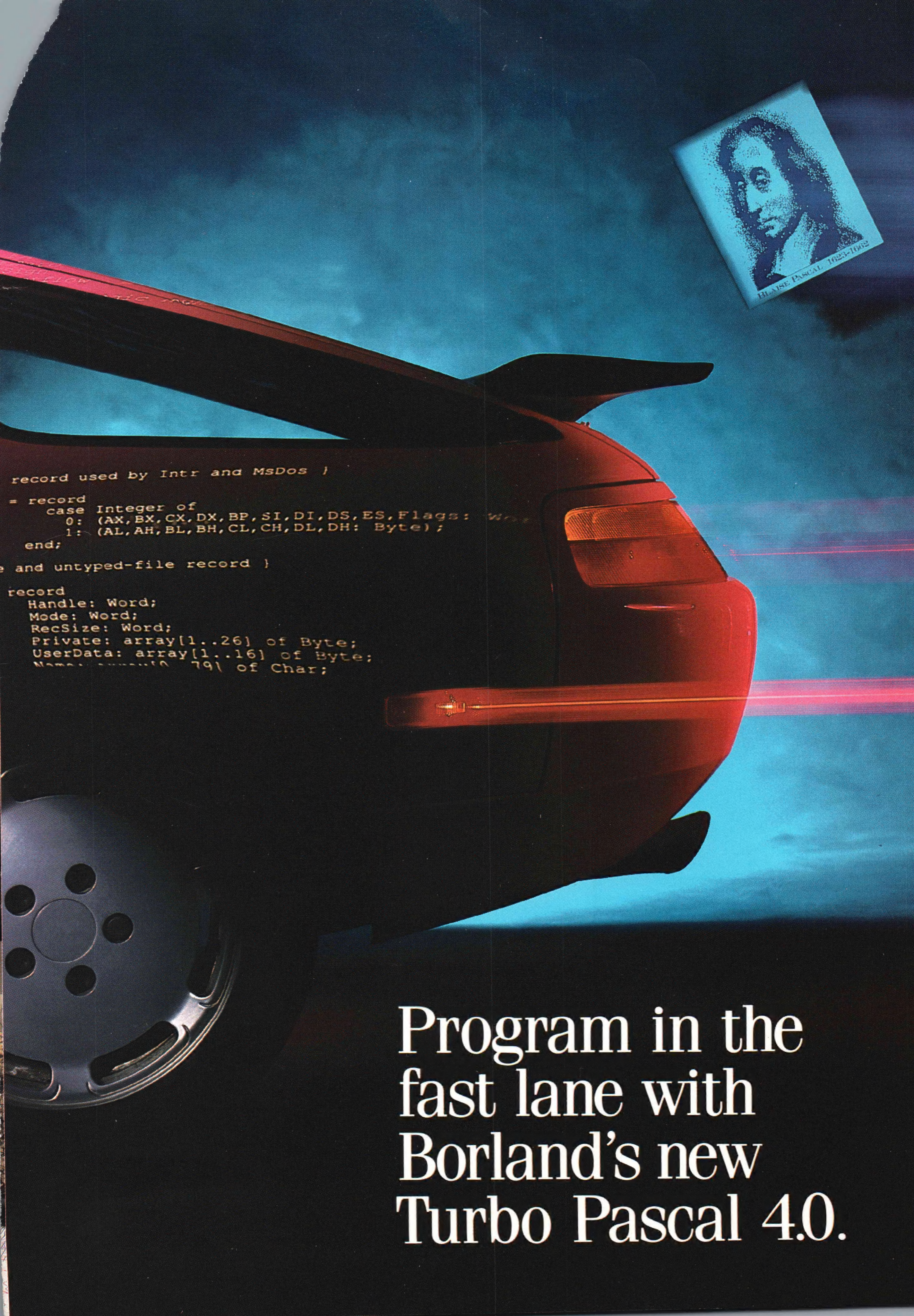
Credit card expiration date: ______/______

Card # |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

*In US please add $5 shipping for each product ordered or $15 for the Compiler and Developer's Library. Outside US please add $10 shipping and handling for each product ordered or $25 for the Compiler and Developer's Library.

†To qualify for the upgrade price you must give the serial number of the equivalent product you are upgrading.




BLAISE PASCAL 1623-1662
Program in the fast lane with Borland's new Turbo Pascal 4.0.

```
 record used by Intr and MsDos }

 = record
     case Integer of
       0: (AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags: Wo
       1: (AL,AH,BL,BH,CL,CH,DL,DH: Byte);
   end;

e and untyped-file record }

 record
    Handle: Word;
    Mode: Word;
    RecSize: Word;
    Private: array[1..26] of Byte;
    UserData: array[1..16] of Byte;
    Name: array[0..79] of Char;
```

The fast lane is *fa*

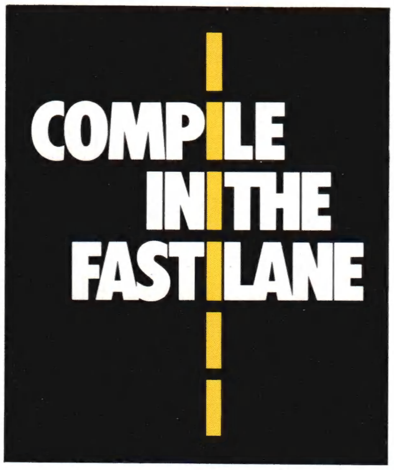
Our new Turbo Pascal® 4.0 is so fast, it's almost reckless. How fast? Better than 27,000 lines of code per minute. That's much faster than 3.0 or any other Pascal compiler and the reason why you need 4.0 today.

Pascal. The fastest and the best.

If you're just now learning a computer language, learn Pascal. If you're already programming in Pascal, you're programming with a winner because Pascal is the worldwide language of choice. Pascal is the most popular language in university computer science classes and with computer enthusiasts who appreciate Pascal's modern programming structure. It's powerful, coherent, easy to learn and use—and with Turbo Pascal 4.0—faster than ever before.

Turbo Pascal: Technical excellence

Commitment to technical excellence and



superiority also means commitment to detail, however painstaking, and that takes time. 4.0's predecessor, Turbo Pascal 3.0 is the worldwide standard, and with Turbo Pascal 4.0, we've bettered that standard. 4.0 is clearly the world's fastest development tool for the IBM® PS/2 series, PC's and compatibles—and the world's favorite Pascal compiler.

4.0 breaks the code barrier

No more swapping code in and out to beat the 64K code barrier. Designed for large programs, Turbo Pascal 4.0 lets you use every byte of memory in your computer. You paid for all that memory, now you can use it freely.

For the IBM PS/2 and the IBM and Compaq families of personal computers and all 100% compatibles.

YES! **I want to upgrade to Turbo Pascal 4.0 and the 4.0 Toolboxes**

Registered owners have been notified by mail. If you are a registered Turbo Pascal user and have not been notified of Version 4.0 by mail, please call us at *(800) 543-7543*. To upgrade if you have not registered your product, just send the original registration form from your manual and payment with this completed coupon to:

Pascal 4.0 Upgrade Dept.
Borland International
4585 Scotts Valley Drive
Scotts Valley, CA 95066

Name ______________________________
Ship Address ______________________________
City ______________________ State __________
Zip ______________ Telephone (   ) __________

This offer is limited to one upgrade per valid registered product. It is good until November 30, 1987. Not good with any other offer from Borland. Please allow 4 to 6 weeks for delivery of Toolboxes.

Outside U.S. make payments by bank draft payable in U.S. dollars drawn on a U.S. bank. CODs and purchase orders will not be accepted by Borland.

Now's the time for a *fast* decision: Upgrade now to 4.0!

Compatibility with Turbo Pascal 3.0

We've created 4.0 to be highly compatible with version 3.0 and included a conversion program and compatibility units to help you convert all your 3.0 programs to 4.0.

Highlights of Borland's new Turbo Pascal 4.0

- Compiles 27,000 lines per minute
- Supports >64K programs
- Uses units for separate compilation
- Integrated development environment
- Interactive error detection/location
- Includes a command line version of the compiler

4.0 also

- Saves output screen in a window
- Supports 25, 43 and 50 lines per screen
- Generates MAP files for debugging
- Has graph units including CGA, EGA, VGA, MCGA, 3270 PC, AT & T 6300 & Hercules support
- Supports extended data types (including word, long integers)
- Does smart linking
- Comes with a free revised MicroCalc spreadsheet source code

4.0 is all yours for only $99.95

Sieve (25 iterations)

	Turbo Pascal 4.0	***Turbo Pascal 3.0***
Size of Executable File	2224 bytes	11682 bytes
Execution speed	9.3 seconds	9.7 seconds

Sieve of Eratosthenes, run on an 8MHz IBM AT

Since the source file above is too small to indicate a difference in compilation speed we compiled our GOMOKU program from Turbo Gameworks to give you a true sense of how much faster 4.0 really is!

Compilation of GO.PAS (1006 lines)

	Turbo Pascal 4.0	***Turbo Pascal 3.0***
Compilation speed	2.2 seconds	3.6 seconds
Lines per minute	27,436	16,750

GO.PAS compiled on an 8 MHz IBM AT

60-Day Money-Back Guarantee**

BORLAND
INTERNATIONAL

For the dealer nearest you or to order call
(800) 543-7543.

BORLAND

CIRCLE 161 ON READER SERVICE CARD

More Magic from Blaise.

Turbo C TOOLS™

Magic is easy with Turbo C TOOLS in your bag of tricks. New Turbo C TOOLS™ from Blaise Computing is a library of compiled C functions that allows you full control over the computer, the video environment, and the file system, and gives you the jump on building programs with Borland's new C compiler. Now you can concentrate on the creative parts of your programs.

The library comes with well-documented source code so that you can study, emulate, or adapt it to your specific needs. Blaise Computing's attention to detail, like the use of function prototyping, cleanly organized header files, and a comprehensive, fully-indexed manual, makes Turbo C TOOLS the choice for experienced software developers as well as newcomers to C.

Turbo C TOOLS provides the sophisticated, bullet-proof capabilities needed in today's programming environment, including removable windows, "sidekickable" applications, and general interrupt service routines written in C.

The functions contained in Turbo C TOOLS are carefully crafted to supplement Turbo C, exploiting its strengths without duplicating its library functions. As a result you'll get functions written predominantly in C, that isolate hardware independence, and are small and easy to use.

Turbo C TOOLS embodies the full spectrum of general purpose utility functions that are critical to today's applications. Some of the features in Turbo C TOOLS are:

◆ **WINDOWS** that are stackable and removable, that have optional borders and a cursor memory, and that can accept user input.

◆ **INTERRUPT SERVICE ROUTINE** support for truly flexible, robust and polite applications. We show you how to capture DOS critical errors and keystrokes.

◆ **INTERVENTION CODE** lets you develop memory resident applications that can take full advantage of DOS capabilities. With simple function calls, you can schedule a Turbo C function to execute either when a "hot key" is pressed or at a specified time.

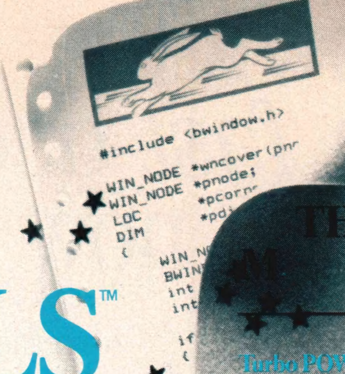
◆ **RESIDENT SOFTWARE SUPPORT** lets you create, detect, and remove resident utilities that you write with Turbo C TOOLS.

◆ **FAST DIRECT VIDEO ACCESS** for efficiency, and support for all monitors including EGA 43-line mode.

◆ **DIRECTORY AND FILE HANDLING** support let you take advantage of the DOS file structure, including volume labels and directory structure.

In addition to Turbo C TOOLS, Blaise Computing Inc. has a full line of support products for Microsoft, Lattice and Datalight C, Microsoft Pascal and Turbo Pascal. Call today for details, and make magic!

Turbo C TOOLS supports the Borland Turbo C compiler, requires DOS 2.00 or later and is just $129.00



```
#include <bwindow.h>

WIN_NODE *wncover(pnr
WIN_NODE *pnode;
LOC      *pcorne
DIM      *pdi
{
    WIN_N
    BWIN
    int
    int

    if
    {
```

THE BLAISE MENU

Turbo POWER TOOLS PLUS *$99.95*
Screen and window management including EGA support; DOS memory control; ISRs; scheduled intervention code; and much more. For Turbo Pascal.

Turbo POWER SCREEN
COMING SOON! General screen management; paint screens; block mode data entry or field-by-field control with instant screen access. For Turbo Pascal.

Turbo ASYNCH PLUS *$99.95*
Interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem and XMODEM control. For Turbo Pascal.

PASCAL TOOLS/TOOLS 2 *$175.00*
Expanded string and screen handling; graphics routines; memory management; general program control; DOS file support and more. For MS-Pascal.

C TOOLS PLUS *$175.00*
Windows; ISRs; screen handling; multiple monitors; EGA 43-line text mode; direct screen access; DOS file handling and more. For MS and Lattice C version 3.00 and later.

LIGHT TOOLS *$99.95*
Windows; ISRs; EGA 43-line text mode; direct screen access; DOS file handling and more. For the Datalight C compiler.

ASYNCH MANAGER *$175.00*
Full featured interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem control and XMODEM. For C or MS-Pascal.

VIEW MANAGER *$275.00*
General screen control; paint screens; block mode data entry or field-by-field control with instant screen access. For C or MS-Pascal.

RUNOFF *$49.95*
Text formatter for all programmers; flexible printer control; user-defined variables; index generation; general macro facility. Crafted in Turbo Pascal.

EXEC *$95.00*
NEW VERSION! Program chaining executive. Chain one program from another in different languages; specify common data areas; less than 2K of overhead.

**TO ORDER CALL TOLL FREE
800-333-8087
TELEX NUMBER-338139**

YES! I want to make magic!
Enclosed is $________ for ________ copies of ________.
☐ Please send me more information on your products.
CA residents add Sales Tax. Domestic orders add $4.00 for UPS shipping, $10.00 for Federal Express standard air.
Name: ________ Phone: (____) ________
Address: ________
City: ________ State: ________ Zip: ________
VISA or MC#: ________ Exp. Date: ________

Turbo C is a trademark of Borland International.

BLAISE COMPUTING INC.
2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

CIRCLE 159 ON READER SERVICE CARD

NOVEMBER 1987

CONTENTS

VOLUME 12, ISSUE 11

ARTICLES

Mac graphics ▶
3-D Images from Contour Maps **18**
by William D. May
Bill shows how to warp scanned images of contour maps—such as topographical maps—into representations that can be displayed and rotated in three dimensions.

Turbo C ▶
A Graphics Toolbox for Turbo C (Part 1) **30**
by Kent Porter
Borland's new C compiler doesn't come with a lot of graphics support. Kent decided to do something about that.

Turbo Pascal ▶
A Graphics Toolkit for Turbo Pascal **38**
by Hubert D. Callihan
Hugh uses several nonstandard Turbo Pascal routines to create a set of tools for handling screen regions.

Using EGA Graphics Screens in Your Programs **46**
by J. Brooks Breeden
Using both Forth and pseudocode (for portability), Brooks gives you the tools for loading EGAPaint files into video memory.

Make your own TSRs ▶
Automated Interrupt Handling in C **54**
by Ron Miller
Ron offers a clever hack you can use to develop your own TSR utilities.

Text processing ▶
An Alternative to Soundex **62**
by Jim Howell
Weight or wait—a faster word matching algorithm

Async AppleTalk **104**
Richard E. Brown and Steve Ligett's listings continued from last month

COLUMNS

Unix time ▶
C CHEST **116**
by Allen Holub
Allen shows how to use the ANSI time functions.

STRUCTURED PROGRAMMING **124**
by Namir Clement Shammas
Namir presents several heuristic search techniques.

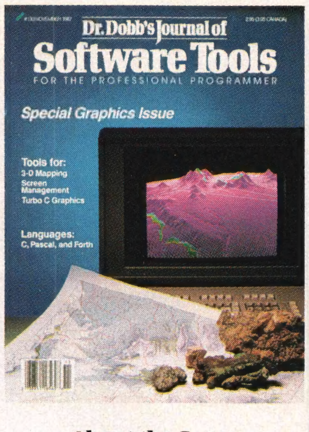
OOPstories ▶
ARTIFICIAL INTELLIGENCE **130**
by Ernest R. Tello
Ernie talks with users of object-oriented programming tools and tells why some developers are abandoning or avoiding the object-oriented approach—and then he tells why he still thinks it's one of the best paradigms yet devised.

FORUM

EDITORIAL **6**
by Michael Swaine
RUNNING LIGHT **8**
by Tyler Sperry
ARCHIVES **8**
LETTERS **12**
by you
SWAINE'S FLAMES **152**
by Michael Swaine

PROGRAMMER'S SERVICES

OF INTEREST: **144**
Products for programmers
ADVERTISER INDEX: **145**
Where to go for more information on products

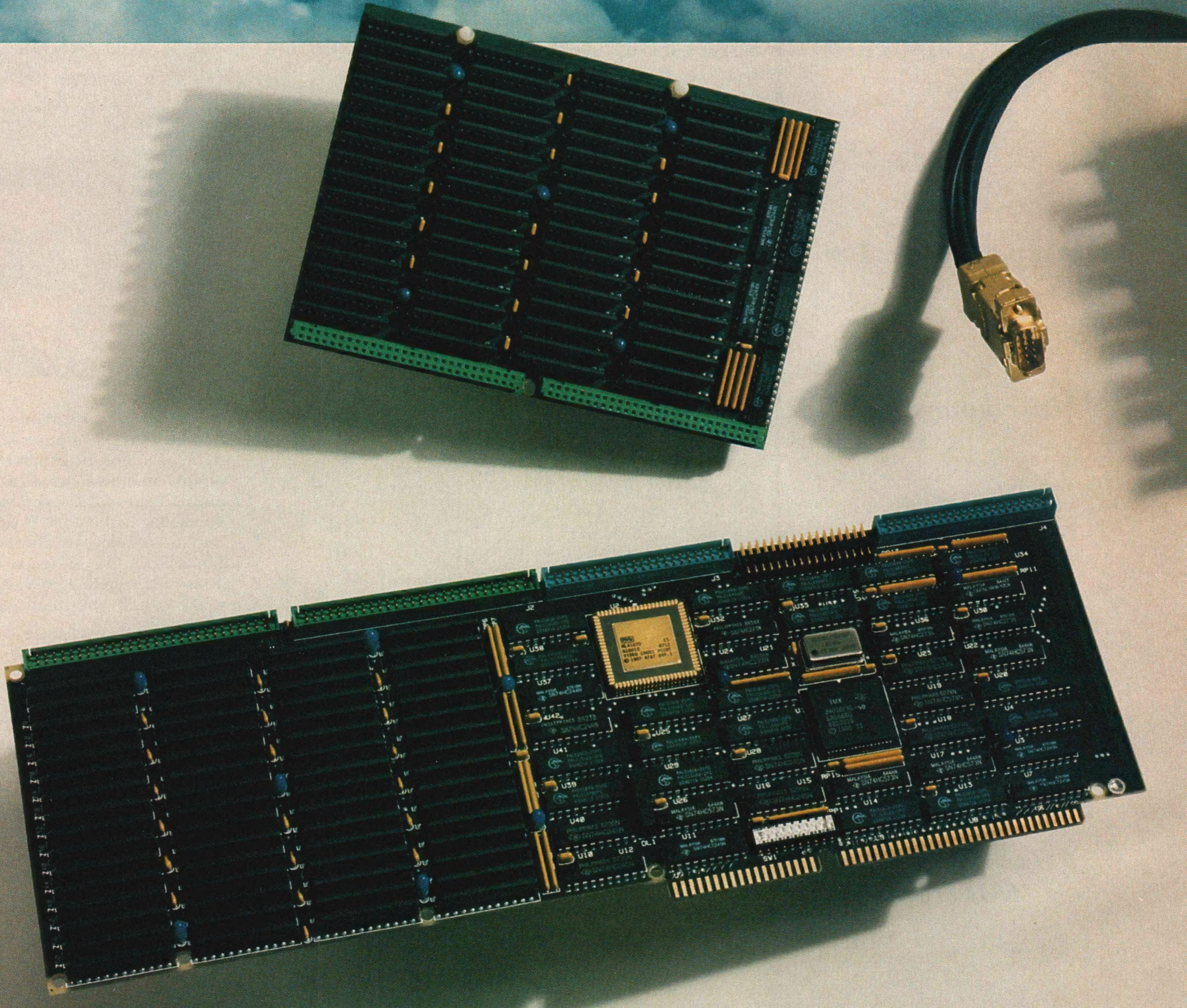


About the Cover
As you may have guessed, we didn't produce the contour map on this month's cover on the art department's 512K Mac. Thanks, you folks at Dynamic Graphics Inc. in Berkeley, California, for the use of the software, Vax, Tektronix terminal, and data to make this month's cover happen.

This Issue
Programming is getting more like cinematography every day. Welcome to our graphics programming issue, with coverage ranging from how to handle contour maps on the Macintosh to how to shuffle screen regions on the PC. There's even a library for handling graphics in Turbo C.

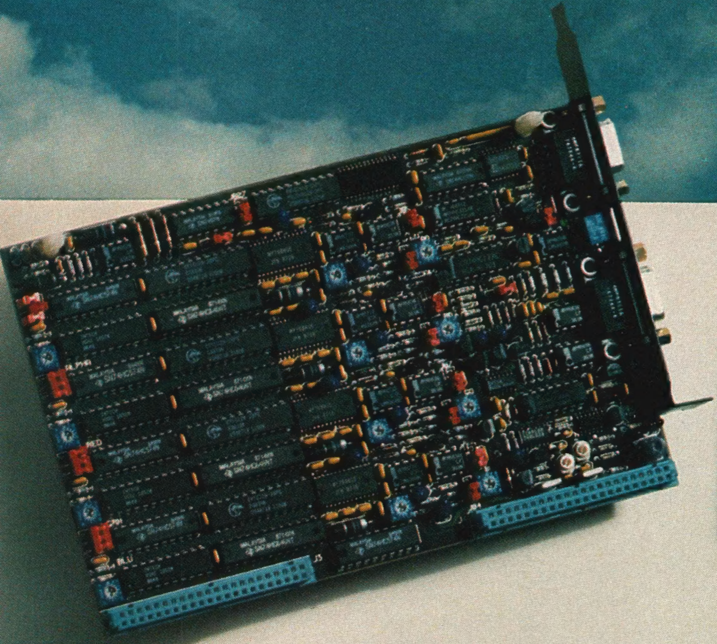
Next Issue
December is our annual operating systems issue. The lead article comes from Dave Cortesi (the Resident Intern, for those of you who remember his column from the old days) and concerns one of the less explored aspects of OS/2: dynamic linking. Allen Holub's column will detail a new multitasking kernel, and we'll also present some keen-edged blades for system hacking.

Now Taking Applications.

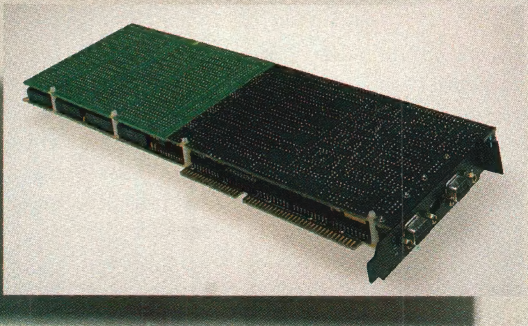


Take a look at the specs on VISTA™, a good look. Notice the processing, programming, and video capabilities? Now think real hard about what *you* could do with the power of VISTA and a microcomputer. Incorporate it with your system to create a digital pre-press proofing station for publishing. Design a graphics workstation which outputs both colorful hi-resolution slides and broadcast-quality animated images. Construct a CAD system which merges computer generated images with real-life backdrops for architecture, packaging or other industries. And, after you've brainstormed your way to new horizons of videographics possibilities, get your own VISTA and start working.

Introducing VISTA™ Videographics.



VISTA is a powerful single-slot videographics adapter which captures and displays video signals in real time.



Let's Get Specific.

We knew you couldn't resist seeing the facts, and frankly, our engineers wouldn't have it any other way. Here is an overview of VISTA's key features.

FEATURES:

- 4Mbytes of Video RAM on-board
- Texas Instruments' TMS 34010 GSP
- Flexible, programmable resolutions
- NTSC and PAL compatible
- Four 8-bit channels for real-time capture
- Fully integrated genlock
- Processor memory expandable in 2Mbyte increments to 12Mbytes
- Four 2K x 8-bit CMOS static RAM LUTs
- Display can be color-mapped, RGB, or a versatile combination of both
- Interlaced and non-interlaced display
- Binary and fractional programmable zoom capability, creates horizontal and vertical magnify or minify
- Smooth horizontal and vertical programmable panning, includes wrap-around and split screen
- Suggested Retail Price: $5995.

ADDRESSABLE RESOLUTIONS:

32 bits/pixel	16 bits/pixel	8 bits/pixel
1024x1024	2048x1024	4096x1024
512x2048	1024x2048	2048x2048
256x4096	512x4096	1024x4096

CAPTURE RESOLUTIONS:*

NTSC	PAL
(RS-170A)	(CCIR-624)
756x486	738x576
604x486	590x576
504x486	492x576
432x486	422x576

Resolutions are programmable; these are nominal ones for interlaced NTSC and PAL compatible.

DISPLAY RESOLUTIONS:*

NTSC	PAL	Interlaced	Non-Interlaced
(RS-170A)	(CCIR-624)		
1512x486	1476x576	1024x768	768x576
1008x486	984x576	(60 Hz)	(50 Hz)
756x486	738x576		
604x486	590x576	768x768	756x486
504x486	492x576	(80 Hz)	(60 Hz)

*Resolutions are programmable; these are nominal ones.

COMPUTER REQUIREMENTS:

Host Type:	IBM PC AT and 100% Compatibles, Compaq 386, Apollo DN 3000-single-slot board
Data Bus:	16-bit or 8-bit (self-configuring)
Bus Clock:	6MHz to 12MHz
Power Consumption:	15 Watts

It's So Flexible, We've Added Support.

With its Texas Instruments TMS 34010 graphics processor, large quantity of video memory, and proprietary video cross-point, VISTA can be programmed for an array of powerful market-specific videographic applications. To help you maximize VISTA's potential, Truevision offers a range of C-language programming tools for developers. And when your system is market-ready, we'll support your marketing efforts with our *TRUEVISION SOFTWARE CATALOG, TRUEVISION NEWS,* and *THE PULSE.*

We're For Higher

Resolution…Power…Flexibility… Quality. Join the many key manufacturers and developers already working with the state of the videographics art, VISTA. Call us at **800/858-TRUE** for more information on the VISTA Developer's Program. We're ready to take your application today.

AT&T
Electronic Photography and Imaging Center
7351 Shadeland Station, Suite 100
Indianapolis, IN 46256
800/858-TRUE

AT&T | TRUEVISION®

CIRCLE 294 ON READER SERVICE CARD

EDITORIAL

Mike's Survey

We've just seen the results of our annual reader survey, so we now know more about you and your interests. We know that you are educated: most of you have done graduate work. You're forward-looking: you're more likely to use LISP or PROLOG than COBOL or even Modula-2, and if you or your company don't already have a 386 machine, you probably will buy one or more in the next year. You have professional access to a wide choice of machines, operating systems, and environments for software development.

Not entirely coincidentally, we're just wrapping up the editorial calendar for 1988. You can look forward to coverage of a wide range of programming environments: Unix, OS/2, the Macintosh environment, Windows, and DOS. We'll publish code in PROLOG, LISP, PostScript, C, Pascal, assembly languages, Forth, and BASIC. We'll examine the shift in programming style to object-oriented, event-driven design, we'll . . . .

We'll gladly, as always, alter the plan to accommodate new developments and the treasures of the transom. Which is where you come in.

What follows is a list of topics that interest us, beyond the languages and operating systems mentioned above. Survey results indicate that they also interest you. You can influence us in two ways. The first is to write us an article on one of the listed topics (or another of your choosing). The second is to let us know where you'd like to see us invest our efforts in the coming year. Treat this page as a ballot and check your favorite topics. Send me your choices. I promise to read them.

Ada
Algorithms
Common data formats
Consulting
Databases
Device control
Editors
EMS programming
Encryption
Fourth generation languages
Functional programming
Human interface design
Industry news
Libraries
Logic programming
Machine learning
Managing development teams
Mathematics
Modula-2
Multitasking
Music
Nubus and the Mac II
Numerical methods
PS/2 programming
Parallel processing
PostScript
Product news
Product reviews
Scientific applications
Software engineering
Starting a software business
Telecommunications
Unix

Please indicate your choices on this page or a copy of it and send it to

Michael Swaine
Mike's Survey
M&T Publishing
501 Galveston Drive
Redwood City CA 94063.

[illegible]

Michael Swaine
editor-in-chief

Dr. Dobb's Journal of Software Tools
FOR THE PROFESSIONAL PROGRAMMER

Editorial

Editor-in-Chief	*Michael Swaine*
Editor	*Tyler Sperry*
Managing Editor	*Vince Leone*
Associate Editor	*Ron Copeland*
Assistant Editors	*Sara Noah Ruddy*
Technical Editors	*Allen Holub*
	Richard Relph
Contributing Editors	*Kent Porter*
	Namir Shammas
	Ernest R. Tello
Copy Editor	*Rhoda Simmons*

Production

Director Art/Production	*Larry L. Clay*
Art Director	*Michael Hollister*
Assoc. Art Director	*Joe Sikoryak*
Technical Illustrator	*Frank Pollifrone*
Typesetter	*Mary Lopez*
Cover Photographer	*Michael Carr*

Circulation

Circulation Director	*Maureen Kaminski*
Fulfillment Coordinator	*Francesca Martin*
Book Marketing Mgr.	*Jane Sharninghouse*
Subscription Supervisor	*Kathleen Shay*
Newsstand Sales Coordinator	*Larry Hupman*

Administration

Finance Director	*Kate Wheat*
Business Manager	*Betty Trickett*
Accounts Payable Supv.	*Mayda Lopez-Quintana*
Accts. Receivable Supv.	*Laura Di Lazzaro*

Advertising Director
Ferris Ferdon (415) 366-3600

Marketing Mgr	*Michael Wiener*
Trafficking Coordinator	*Patricia Albert*

Account Managers
see page 145

Associate Publisher
Michael Swaine
Assistant *Sara Noah Ruddy*

Dr. Dobb's Journal of Software Tools (USPS 307690) is published monthly by M&T Publishing Inc., 501 Galveston Dr., Redwood City, CA 94063; (415) 366-3600. Second-class postage paid at Redwood City and at additional entry points. *DDJ* is published under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a nonprofit corporation.

Article Submissions: Send manuscripts and disk (with article and listings) to the Editor.

***DDJ* on CompuServe:** Type GO DDJ

Address Correction Requested: Postmaster: Send Form 3579 to *Dr. Dobb's Journal*, P.O. Box 27809, San Diego, CA 92128. **ISSN 0888-3076**

Customer Service: For subscription problems call: outside CA (800) 321-3333; in CA (619) 485-9623 or 566-6947. For book/software order problems call (415) 366-3600.

Subscriptions: $29.97 per 1 year; $56.97 for 2 years. Canada and Mexico add $27 per year airmail or $10 per year surface. All other countries add $27 per year airmail. Foreign subscriptions must be prepaid in U.S. funds drawn on a U.S. bank. For foreign subscriptions, TELEX: 752-351.

Foreign Newsstand Distributor: Worldwide Media Service Inc., 386 Park Ave. South, New York, NY 10016; (212) 686-1520 TELEX: 620430 (WUI).

Our thanks to NASA for supplying this computer enhanced ultraviolet photo taken by Skylab IV of a solar prominence reaching out 350,000 miles above the sun's surface

Genius Begins With A Great Idea ...

But The Idea Is Just The Beginning

What follows is the time consuming task of giving form and function to the idea.

That's why we concentrate on building into our software development systems functions and features that help you develop your software ideas in less time and with less effort.

We've started 1987 by releasing new versions of our MS-DOS, Macintosh, Amiga, ROM, and Apple // C development systems. Each system is packed with new features, impressive performance, and a little bit more genius.

FREE 2-DAY DELIVERY!

Aztec C86 4.1

New PC/MS-DOS • CP/M-86 • ROM

Superior performance, a powerful new array of features and utilities, and pricing that is unmatched make the new Aztec C86 the first choice of serious software developers.

Aztec C86-p Professional System $199
• optimized C with near, far, huge, small, and large memory + Inline assembler + Inline 8087/80287 + ANSI support + Fast Float (32 bit) + optimization options • Manx Aztec 8086/80x86 macro assembler • Aztec overlay linker (large/small model) • source level debugger • object librarian • 3.x file sharing & locking • comprehensive libraries of UNIX, DOS, Screen, Graphics, and special run time routines.

Aztec C86-d Developer System $299
• includes all of Aztec C86-p • Unix utilities make, diff, grep • vi editor • 6+ memory models • Profiler.

Aztec C86-c Commercial System. $499
• includes all of Aztec C86-d • Source for library routines • ROM Support • CP/M-86 support • One year of updates.

Aztec C86 Third Party Software

A large array of support software is available for Aztec C86. Call or write for information. The following is a list of the most requested products: • Essential Graphics • C Utility Library • Curses • Greenleaf Communication, General, and Data Window • Halo • Panel+ • PC-lint • PforCe • Pre-C • Windows for C • Windows for Data • C terp • db_Vista • db-Query • Phact • Plink-86 Plus • c-tree • r-tree • Pmate

CP/M • TRS-80 • 8080/Z80 ROM

C compiler, 8080/Z80 assembler, linker, librarian, UNIX libraries, and specialized utilities.

Aztec C II-c (CP/M-80 & ROM)	**$349**
Aztec CII-d (CP/M-80)	**$199**
Aztec C80 (TRS-80 3&4)	**$199**

Aztec C68k/Am 3.4

New Amiga Release

Amiga user groups across the USA voted Aztec C68k/Am release 3.3 the best Software Development System for the Amiga. Release 3.4 is more impressive.

Aztec C68k/Am-p Professional $199
A price/feature/performance miracle. System includes: optimized C • 68000/680x0 assembler • 68881 support • overlay linker • UNIX and Amiga libraries • examples.

Aztec C68k/Am-d Developer $299
The best of Manx, Amiga, and UNIX. System includes: all of Aztec C68k/Am-p • the Unix utilities make, diff, grep and vi.

Aztec C68k/Am-c Commercial. $499
Aztec C68k/Am-d plus source for the libraries and one year of updates.

FREE 2-DAY DELIVERY!

Aztec C68k/Mac 3.4

New Macintosh Release

For code quality, reliability, and solid professional features, Aztec C for the Macintosh is unbeatable. This new release includes features and functions not found in any other Macintosh C development system.

Aztec C68k/Mac-p Professional $199
• optimized C • 68000/680x0 assembler • 68881 support • overlay linker • UNIX and Macintosh libraries • examples.

Aztec C68k/Mac-d Developer. $299
The best of Manx, Macintosh, and UNIX. System includes: all of Aztec C68k/Am-p • the Unix utilities make, diff, grep • vi editor.

Aztec C68k/Mac-c Commercial $499
Aztec C68k/Am-d plus source for the libraries and one year of updates.

Aztec C65

New ProDOS Release

Aztec C65 is the only commercial quality C compiler for the Apple II. Aztec C65 includes C compiler, 6502/65C02 assembler, linker, library utility, UNIX libraries, special purpose libraries, shell development environment, and more. An impressive system.

Aztec C65-c Commercial $299
• runs under ProDOS • code for ProDOS or DOS 3.3

Aztec C65-d Developer. $199
• runs under DOS 3.3 • code for DOS 3.3

Aztec ROM Systems

6502/65C02 • 8080/Z80 • 8086/80x86 • 680x0

An IBM or Macintosh is not only a less expensive way to develop ROM code, it's better. Targets include the 6502/65C02, 8080/Z80, 8086/80x86, and 680x0.

Aztec C has an excellent reputation for producing compact high performance code. Our systems for under $1,000 outperform systems priced at over $10,000.

Initial Host Plus Target	**$750**
Additional Targets	**$500**
ROM Support Package	**$500**

Vax, Sun, PDP-11 ROM HOSTS

Call for information on Vax, PDP-11, Sun and other host environments.

C' Prime

PC/MS-DOS • Macintosh Apple II • TRS-80 • CP/M

These C development systems are unbeatable for the price. They are earlier versions of Aztec C that originally sold for as much as $500. Each system includes C compiler, assembler, linker, librarian, UNIX routines, and more. Special discounts are available for use as course material.

C' Prime $75

Aztec Cross Development Systems

Most Aztec C systems are available as cross development systems. Hosts include: PC/MS-DOS, Macintosh, CP/M, Vax, PDP-11, Sun, and others. Call for information and pricing.

How To Become An Aztec C User

To become a user call 800-221-0440. From NJ or international locations call 201-542-2121. Telex: 4995812 or FAX: 201-542-8386. C.O.D., VISA, MasterCard, American Express, wire (domestic and international), and terms are available. One and two day delivery available for all domestic and most international destinations.

Aztec C is available directly from Manx and from technically oriented computer and software stores. Aztec Systems bought directly from Manx have a 30 day satisfaction guarantee.

Most systems are upgradable by paying the difference in price plus $10. Site licenses, OEM, educational, and multiple copy discounts are available.

MANX

CIRCLE 108 ON READER SERVICE CARD

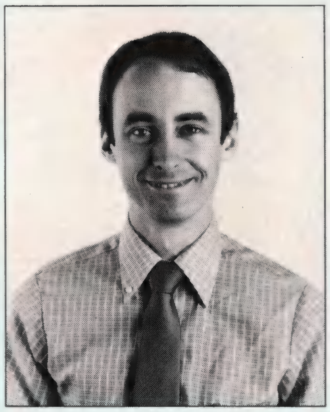
Manx Software Systems
1 Industrial Way, Eatontown, NJ 07724

To order or for more information call today.

1-800-221-0440

In NJ or international call (201) 542-2121 • TELEX: 4995812

RUNNING LIGHT



As Mike Swaine mentioned on page 6, we've just received the results of our latest reader survey. After spending some time pouring through the data, I've still got a few things I'm curious about—items that are important to producing a quality magazine but don't often show up in the statistical abstracts.

For example, we devote a lot of space to source code listings. The question is, are you folks really using those listings? Given there's always a limit to the number of pages in an issue, should we keep printing the listings as they are or print more articles and arrange for source code to be distributed some other way?

There's ample historical justification for printing source listings, of course. The reason *DDJ* was founded, after all, was to put tools such as the source code for Tiny BASIC into the hands of as many people as possible. If you wanted your computer to do something back then, for the most part you had to write the program yourself. (Sometimes you'd have to write an interpreter or compiler first and then get back to the original problem.) A dozen years ago, the hacker ethic demanded we put source code into the hands of our readers, and we did it gladly.

But that was a dozen years ago, and things have changed a bit. These days Bill Gates isn't paranoid about hobbyists copying his cassette BASIC; instead he's justifiably worried about Philippe Kahn stealing his languages market. The economics of the eight million PCs out there (a conservative estimate, by the way) have caused many sophisticated compilers to be priced so that it doesn't make sense to pirate the products, let alone write them yourself.

Fine, I hear you say, but what has all this to do with *Dr. Dobbs?*

Simply put, the field is changing and *DDJ* must grow and adapt to keep up. There's no question that Tiny BASIC ported to the AMD 29000 would be an interesting exercise in assembly language, but surely there are more useful projects to use as an example of 29000 programming. (For those of you coding for the 29000, yes, that was a hint for article suggestions.)

The question is not whether the Doctor's pride and joy will change, but how. To get back to the topic of source listings: Most of you have a modem or easy access to one. In a time when 2,400 bps modems are readily available, does it make sense to print listings so that you can enjoy hours of typing practice? Do you want more downloading options or is CompuServe access sufficient? Finally, do you use the code in binary form or just scan the printed listings to understand the algorithms?

As always, I'm willing to discuss this topic with you by phone, if you can catch me at my desk. I'll certainly be watching the CompuServe DDJ Forum to catch the debate and as the pile of papers hiding my desk will attest, we do read all your letters. But if you really want to make sure I get your vote/suggestion/threatening letter, the best bet is to use the mail systems on CompuServe (#<TK>) or BIX ('tyler').

Tyler Sperry

Tyler Sperry
editor

ARCHIVES

Linguistic Overbyte

"Programming languages appear to be in trouble. Each successive language incorporates, with a little cleaning up, all the features of its predecessors plus a few more . . . . The Department of Defense has current plans for a committee-designed language standard that could require a manual as long as 1000 pages. Each new language claims new and fashionable features, such as strong typing or structured control statements, but the plain fact is that few languages make programming sufficiently cheaper or more reliable to justify the cost of producing and learning to use them . . . .

For twenty years programming languages have been steadily progressing toward their present condition of obesity; as a result, the study and invention of programming languages have lost much of their excitement. Instead, it is now the province of those who prefer to work with thick compendia of details rather than wrestle with new ideas."—*John Backus, 1977 Turing Award Lecture.*

Ten Years Ago in *DDJ*

"Microsoft's BASIC for the 8080 and Z-80 . . . is now generally available on both a single-copy and OEM basis. The BASIC became the subject of extended legal dispute which resulted in the termination of an exclusive license to MITS, Inc.

"The BASIC, best known in the field as Altair BASIC, has been in use for 1½ years and has a user base of over 5000."—*news release,* DDJ, *November/December 1977.*

"In his letter in the *DDJ* #18, Phil Karn mentions problems regarding semi-transparent paper tape. I wonder if he has tried using polarized light. Two inexpensive pieces of polarizing material, one above the tape and one below, might solve the problem. Light would still filter through the tape, but would be de-polarized in the process and would be rejected by the second polarizer. Light passing through the holes would still be polarized and be picked up by the sensors. I haven't tried it, but it should work"—*Jim Day, letter to the editor,* DDJ, *November/December 1977.*

DR. DOBB'S JOURNAL of COMPUTER Calisthenics & Orthodontia
Running Light Without Overbyte

Optimizing ROMable C

With Source Level Debugging on Your Target System!

Optimum-C, the compiler known for its tight, highly-optimized code generation, produces completely ROMable code. Coupled with ROM support tools like ROMable standard function libraries, a remote debugger, pre-written startup routine, interactive editor, and UNIX-style MAKE, Optimum-C will help you complete your project in record time.

Optimum-C, the ROM developer's choice

Optimum-C has the features ROM developers require in a C compiler. These features include fast execution, multiple memory support, and 8087 support via inline code or emulation. Optimium-C also has understandable error messages, one-step compiling, complete function prototyping, and inline 8086 I/O instructions.

You've seen the ads: Datalight challenges Microsoft. Our C compiler expert Richard Relph saw the ads and sent for Datalight's compiler. What he found when he began to test it must have given him mixed feelings. For the past two years Richard has been involved in developing the DDJ suite of benchmarks for C compilers. The Datalight compiler flattened those benchmarks, making them worthless.

With the apparent glut of C complier suppliers vying for the MS-DOS market, it was only a matter of time before one of them decided to step above the crowd and provide a reliable optimizing C compiler. Datalight beat all others to the punch by delivering such a compiler February of this year. (Dr. Dobbs Journal, August 1987)

ROM-*it*, The ROM Developer's Kit

ROM-*it* adds the functionality required by serious ROM developers. **ROM-*it*** includes pre-written start-up code for an embedded system to take the 8086 processor from a system RESET to executing C code. The library included has standard C functions that are usable in a ROM environment, without any hidden calls to MS-DOS or the BIOS. Lastly, the BLAZE Locator/Intel hex file generator performs location of code and data while it checks for ROM overflows, illegal data access, and complete program location.

Remote-DSD Source Level debugging on your target system!

The Remote-DSD debugger allows you to debug your application on the target system. Remote-DSD runs on the PC and is connected to your target system via an RS232 link. With Remote-DSD you download your application to the target hardware, modify initial values of variables before you start your application. The C source code appears on the PC showing you where execution will start. You can now execute or single step through your program, set break points, and access your target machine internals like I/O ports, registers, and memory.

And lastly, Support!

The product is only as good as the support. With DATALIGHT, you are covered here. You will get one hour of phone support for those needed answers. Also, you will get updates for one year, so you are always running with the latest versions.

Pick up the phone and gain a powerful advantage.

Call
1-(800) 221-6630

for complete details on how you can finish those ROM projects on time and without the bugs!

Magazine Reviewers Shocked by DATALIGHT's Performance...

"Reviewing this compiler was quite a surprise for us. For such a low price, we were expecting a "lightweight" compiler. What we got was a package that is as good as or better than most of the "heavy-weights." Datalight C implements a complete C language. It also compiles quickly, doesn't take up much disk space, and looks impressive in the benchmarks."
DR. DOBBS, August 1986

"This is a sharp compiler!... what is impressive is that Datalight not only stole the compile time show completely, but had the fastest Fibonacci executable time and had excellent object file sizes to boot!"
COMPUTER LANGUAGE, February 1986

Specifications

- Full UNIX V C compiler with ANSI extensions
- Global optimizations using Data Flow Analysis
- 8087/80287 support inline or emulation
- Complete library source code
- Multiple memory model support
- Interrupt handling in C
- Make utility
- EZ editor
- DLC one step compiling
- Start-up code for 8086
- ROMable library (without hidden MS-DOS calls)
- BLAZE locator/Intel hex file generator
- Source level debugging on the target system
- View source code as it executes on target
- Access local variables
- View 8086 machine internals

Datalight

17505-68th Avenue NE, Suite 304
Bothell, Washington 98011 USA
(206) 367-1803

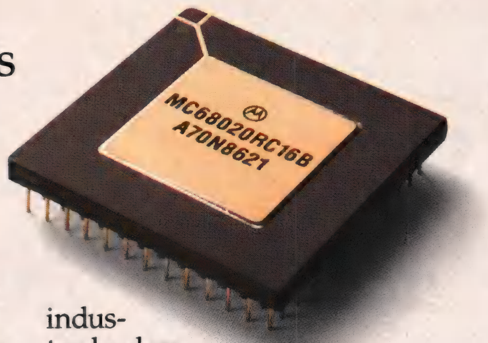
CIRCLE 203 ON READER SERVICE CARD

MOTOROLA

File Edit Scaling CLUT Convert Special
motor
68020
Macintosh II

Apple picked our brains.

And so did hundreds of other companies.



Before millions of people picked Macintosh,™ Apple® picked Motorola's M68000 Family—the brains behind one of the most successful computer products ever launched.

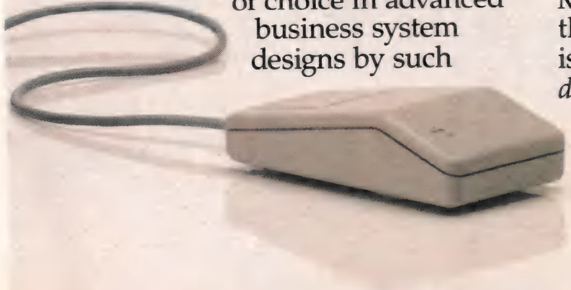
Now Apple has tapped the brainpower of the Motorola MC68020 microprocessor for the Macintosh II, bringing the high performance of a graphics workstation to business desktops everywhere.

72% of all 32-bit systems ever shipped included at least one MC68020. That's more than half a million high-performance systems.

The high-performance business solution.

The MC68020 is not just the overwhelming choice in workstations–it is now setting new performance standards in the office–where it is essential to the computation, graphics and communication necessary for interconnected systems.

While Apple's choice of the MC68020 was a smart move, there's no license on genius: the '020 is the microprocessor of choice in advanced business system designs by such industry leaders as Altos, Alpha Micro, Casio, C.Itoh, Fujitsu, Honeywell Bull, NEC, NCR, Olivetti, Plexus, Ricoh, Sanyo, Sharp, TI, Toshiba and UNISYS.



The graphics solution.

The M68000 family helped Apple implement the visionary "point and click" graphic workstyle that has driven productivity up while driving training costs way down. Businesses of all sizes are discovering dramatic productivity increases in office computing through innovations such as desktop publishing.

The software solution.

Among programmers and designers dedicated to creating the best, most innovative applications, the M68000 architecture has been the leading choice by far–with over *seven million M68000* systems installed since 1979.

Meanwhile, the MC68020, on the market now for three years, is already backed by *two billion dollars worth* of 32-bit software. This is more 32-bit software than all competitive products combined!

The Brain Trust: Where M68000 microprocessors predominate.

Engineering Workstations
Apollo, Hitachi, HP, Sony, Sun, Tektronix.
Laser Printers
Apple, Canon, HP, IBM, QMS, Ricoh.
Departmental Computers
Convergent Technologies, Fujitsu, Honeywell Bull, NEC, NCR, UNISYS.
PBX and Telephone Systems
AT&T, Northern Telecom, Siemens.
Fault Tolerant Systems
IBM, NCR, Nixdorf, Stratus, Tandem.
Supercomputers
Alliant, BBN, Caltech, Fifth Generation.
Factory Automation
Allen-Bradley, ASEA, Bailey Controls, GM, Mitsubishi, Square D.

Join the Brain Trust.

Challenge us to persuade you of the sound business and technical reasons to join the M68020 Brain Trust. Write to us at Motorola Semiconductor Products Inc., P.O. Box 20912, Phoenix, AZ 85036.

We're on your design-in team.

Apple is a registered trademark and Macintosh is a trademark of Apple Computer, Inc.

MOTOROLA

LETTERS

8088 Optimization Techniques

Dear *DDJ*,
I enjoyed Tom Disque's article on 8088 optimization techniques (July 1987) but must point out a simple improvement to his Example 1.

He uses a *jcxz* jump to guard against counter *cx* entering a *rep movsw* with a value of 0. Although this would be a necessary prelude to an ordinary loop, where *cx* is tested at the bottom after decrementation, it is wasted before a *rep* prefix. As explained in *The 8088 Book* by Russell Rector and George Alexy (pages 4–46), a check for *cx* = 0 is the very first step in a *rep* instruction, and when it is true initially, the string primitive is not executed.

Particularly for anyone whose use of the computer reflects the name the French give to it—*L'ordinateur,* a machine for putting things in order—the string primitive instructions are among the 8086 family's most distinct advantages and such situations arise frequently. Bearing in mind this behavior of the *rep* prefix and the *cx* register can save both time and memory as well as giving programmers one less detail to worry about.

Paul R. Emmons
438 W. Chestnut St.
West Chester, PA 19380

Dear *DDJ*,
I read Tom Disque's article in the July issue, on 8088 optimizing techniques, with great interest.

As a Motorola-biased programmer, I found it interesting to note that Mr. Disque's 68000 code suffered from an Intel-type restriction. The code itself was very good, but it limited the amount of memory that could be moved to a maximum of between 64K and 256K, depending on the alignment of the memory. Although Mr. Disque used *long*-size instructions when manipulating the length registers, the *dbf* instruction could only loop a maximum of 64K times.

The *dbf* instruction can easily be extended to cope with 32-bit loops by following it with the instructions:

```
sub.1 #$10000,<register>
bcc.s <loop_label>
```

In Mr. Disque's routine, the solution is to replace the branch to exit after *longloop* with the lines:

```
sub.1 #$10000,d0
bcs.s exit
bra.s longloop
```

and to insert the lines:

```
sub.1 #$10000,d0
bcc.s lngloop2
```

after the second *dbf.* The last *dbf* does not need modification as it only ever moves 14 bytes at most.

With these changes Mr. Disque's memory-move routine can then cope with the complete addressing range of all the 680x0 processors.

Andrew Pennell
The Old School, Greenfield
Bedord, MK45 5DE
England

Legacy of the Teletype

Dear *DDJ*,
In the December 1986 Letters column there was a letter from a Mr. Hawkins in which he was complaining among other things about the terseness of the C language. He attributed this "propensity for the least possible typing" to the authors of the language being two fingered typists. More likely it was what they were typing on that inspired them to opt for the least possible typing. C is an older language than many realize since although it was born in the early 1970s along with Unix, it did not escape from its birthplace, the Bell Laboratories, until around 1980. The standard terminal in use back in the early seventies was the teletype machine. In many operating systems from CP/M to Unix the abbreviation TTY, used for things related to the terminals, makes sense only if it is realized that the terminals were originally teletype machines.

The teletype machine was an electromechanical device, not an electronic device, and was therefore very slow by today's standards. Typically it operated at 110 baud or ten characters per second. Furthermore, after a key was struck you had to wait a full tenth of a second before striking the next key or it would be ignored. Since most people, especially when thinking at the keyboard, tend to

Jared's wardrobe—like his programming—isn't EGA compatible.

(continued on page 138)

TURBO PASCAL AND TURBO C... MEET TURBOHALO

"Ideal! TurboHALO does the job comparable to packages costing $3000 to $4000."
Jim Bromley
Superintendent of Spectrum Management

"I like the speed of TurboHALO...it's ten times faster than the competition."
Deniz Terry
Doctoral Candidate

"TurboHALO is so fun... I use it to design programs as a hobby...It's got lots of ability."
William Porter
Control Systems Manager

"We evaluated all of the graphic development packages for Turbo Pascal, and TurboHALO was the hands down winner!"
Quinn Curtis
Largest New England Distributor

It's time to put graphics into your programming.
A picture's worth a thousand words. So your programming isn't complete until you have graphics. TurboHALO brings your screen and printer to life with subroutines that draw, chart, map, and display. All with color, shape, clarity, perspective and motion. With TurboHALO, create any picture you can imagine.

TurboHALO gives you graphics power.
TurboHALO gives you everything you need for Turbo C and Turbo Pascal graphics programming. A library of over 150 graphics subroutines. Drivers for over 42 graphics hardware devices for the IBM PC family and compatibles. You can create the images you want, on the hardware you have!

Fast, proven and reliable.
TurboHALO is up to ten times faster than other graphics toolkits. And with TurboHALO you get proven, reliable programming tools used by professionals for years.

You'll like TurboHALO or your money back!
TurboHALO is available for only $95. You get an unconditional 45-day money-back guarantee on TurboHALO. To order, call your dealer or IMSI at (415) 454-7101 or (800) 222-4723; in CA (800) 562-4723; in Washington DC (202) 363-9340 or in NC (919) 854-4674.

YES, *I want to see the difference TurboHALO makes in my graphics programming! Rush me the following TurboHALO Graphics Toolkit(s) @ $95 each plus $3 shipping. California residents add 6% sales tax.*

☐ *TurboHALO for Turbo Pascal*
☐ *TurboHALO for Turbo C*
☐ *Check enclosed for $ ________ (made payable to IMSI)*
☐ *Charge my credit card for $ ________* ☐ *VISA* ☐ *MasterCard*

Signature

Card Number Expiration Date

Name

Title

Company

Street

City State Zip

Imsi™

DD1187

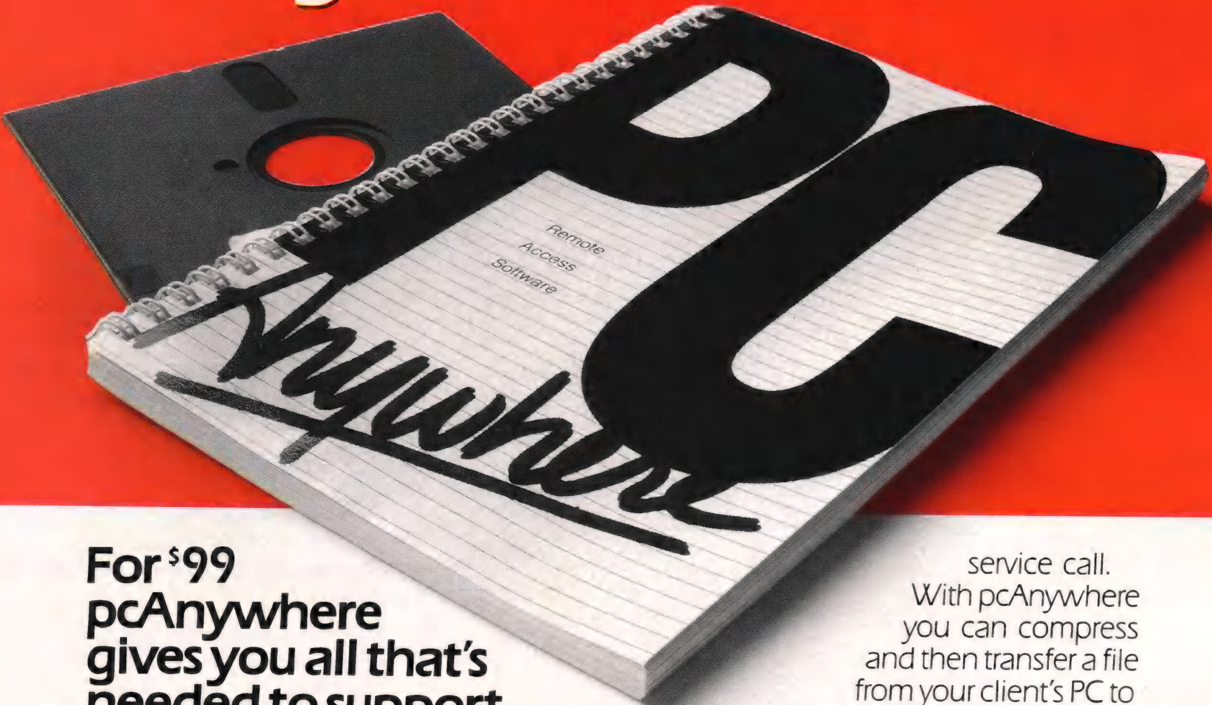
Mail to:
IMSI, 1299 Fourth Street, San Rafael, CA 94901

FOR GRAPHICS PROGRAMMING

TurboHALO requires 256K memory (min); memory resident drivers require 2K (Turbo Pascal only); DOS version 2.0 or higher; Borland language/compiler required. TurboHALO is a trademark of Media Cybernetics and IMSI. Turbo C and Turbo Pascal are trademarks of Borland.

CIRCLE 293 ON READER SERVICE CARD

Support your clients without leaving your office!



For $99 pcAnywhere gives you all that's needed to support both host and remote!

It lets you access and run an unattended IBM PC/XT/AT/PS2 or compatible from any remote location. On any personal computer. Day or night.

And with our new IRMA interface you can access a mini or mainframe, too.

This makes it ideally suited to remote customer service and technical support.

Because you can identify and solve problems from your office PC! Saving valuable time and the expense of a field service call.

With pcAnywhere you can compress and then transfer a file from your client's PC to yours, fix it, then transfer the file back to the client.

Virtually any program on the client's PC can be controlled from your keyboard. As though you were sitting in front of the client's computer.

Take remote control then give it back to your client's PC. Or keep both keyboards active at the same time and type messages back and forth.

Call today for more information or to place an order. You'll find yourself in remote control before you know it!

Visit us at Comdex booth #2027

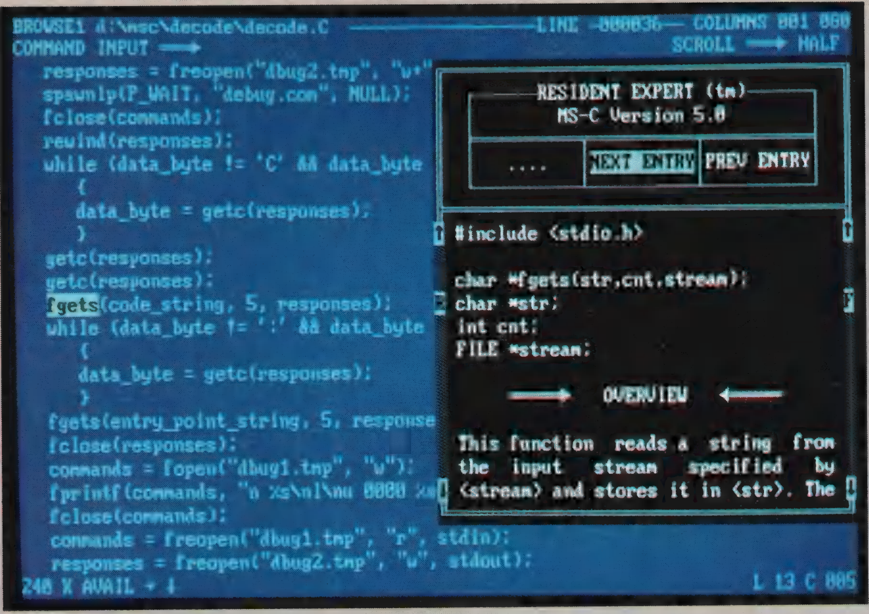
pcAnywhere is a trademark of DMA.

EKD

764 Middle Country Road, P.O. Box Y, Selden, NY 11784 • **(516) 736-0500**

CIRCLE 295 ON READER SERVICE CARD

RESIDENT EXPERT Pop-up Reference Guides...



```
BROWSE1 d:\msc\decode\decode.C ——————————LINE -000036— COLUMNS 001 080
COMMAND INPUT ——→                                        SCROLL ——→ HALF
   responses = freopen("dbug2.tmp", "w+"
   spawnlp(P_WAIT, "debug.com", NULL);
   fclose(commands);
   rewind(responses);
   while (data_byte != 'C' && data_byte
      {
      data_byte = getc(responses);
      }
   getc(responses);
   getc(responses);
   fgets(code_string, 5, responses);
   while (data_byte != ':' && data_byte
      {
      data_byte = getc(responses);
      }
   fgets(entry_point_string, 5, response
   fclose(responses);
   commands = fopen("dbug1.tmp", "w");
   fprintf(commands, "n %s\nl\nu 0000 %
   fclose(commands);
   commands = freopen("dbug1.tmp", "r", stdin);
   responses = freopen("dbug2.tmp", "w", stdout);
240 K AVAIL + ↓                                                   L 13 C 005
```

```
RESIDENT EXPERT (tm)
MS-C Version 5.0

#include <stdio.h>

char *fgets(str,cnt,stream);
char *str;
int cnt;
FILE *stream;

This function reads a string from
the input stream specified by
<stream> and stores it in <str>. The
```

Try One And Get Our MS-DOS/PC-DOS Guide Absolutely FREE!

THE POP-UP REFERENCE REVOLUTION BEGINS

How much development time could you save if you never had to open another PC language or technical reference manual again? What if you could just point at a compiler keyword, assembly instruction, or function name *on your screen* and with a keystroke have complete, authoritative information about language syntax, operands, parameters, examples, and much more?

INTRODUCING THE RESIDENT EXPERT SYSTEM

A growing library of comprehensive, disk resident reference guides about the PC *and* your favorite PC languages. All available instantly through our unique memory resident pop-up access system.

VIRTUALLY EVERYTHING YOU NEED TO KNOW

Each of our *Compiler Reference Guides* contains virtually everything you need to know to program with your *preferred implementation* of your favorite language. Language syntax, all library functions, compiler directives, and error codes are thoroughly documented.

Our *PC Programmer's Reference Guide* documents every PC (and AT) processor instruction and every BIOS and DOS service interrupt. You'll also find tables of keyboard codes, line drawing, ASCII, and IBM character sets, and much more.

THE SPECIALIST'S LIBRARY

Your compiler is unique. That's why our reference guides are *specialized*...each one designed for a particular vendor's language implementation.

Free With Any Purchase!

Our Companion Pop-up Guide To MS-DOS 3.2/PC-DOS 3.3

Limited Time Offer

QUICK DRAW ACCESS SYSTEM

Point-and-shoot...just place the cursor over any term on your screen. Chances are we've got it fully detailed in one of our data bases.

Fully cross indexed...if the instruction or library function you're using isn't quite right, our related topics cross index can help you find a better one.

Multiple volumes on line...you can have one or a dozen of our pop-up reference guides on line...a complete library available instantly.

THE INFORMATION YOU NEED...WHERE YOU NEED IT

Our pop-up shell varies its size and shape dynamically, only taking as much space on your screen as it needs and it *never* covers your working area. You can see your work and our reference data at the *same* time.

A COMPLETE LIBRARY...STILL ONLY A BEGINNING

At Santa Rita, our commitment is to provide the most accurate, extensive selection of PC language reference materials available. If you don't see one of our guides for your favorite language or compiler listed below don't worry, we're probably working on it!

PC Programmer's Reference Guide . . . *(with Assembly Language Guide)*	$59.00
Borland Turbo C *(1.0)*	59.00
Borland Turbo Pascal *(3.0 and below)* . . . . . *(with Graphics & Numerical Methods Toolbox)*	59.00
Borland Turbo Prolog *(1.1 and below)* . . . . . *(with Prolog Toolbox)*	59.00
Lattice C Compiler *(3.2 and below)*	59.00
Mark Williams LetsC *(4.0 and below)*	59.00
Microsoft C Compiler *(5.0 and below)*	59.00

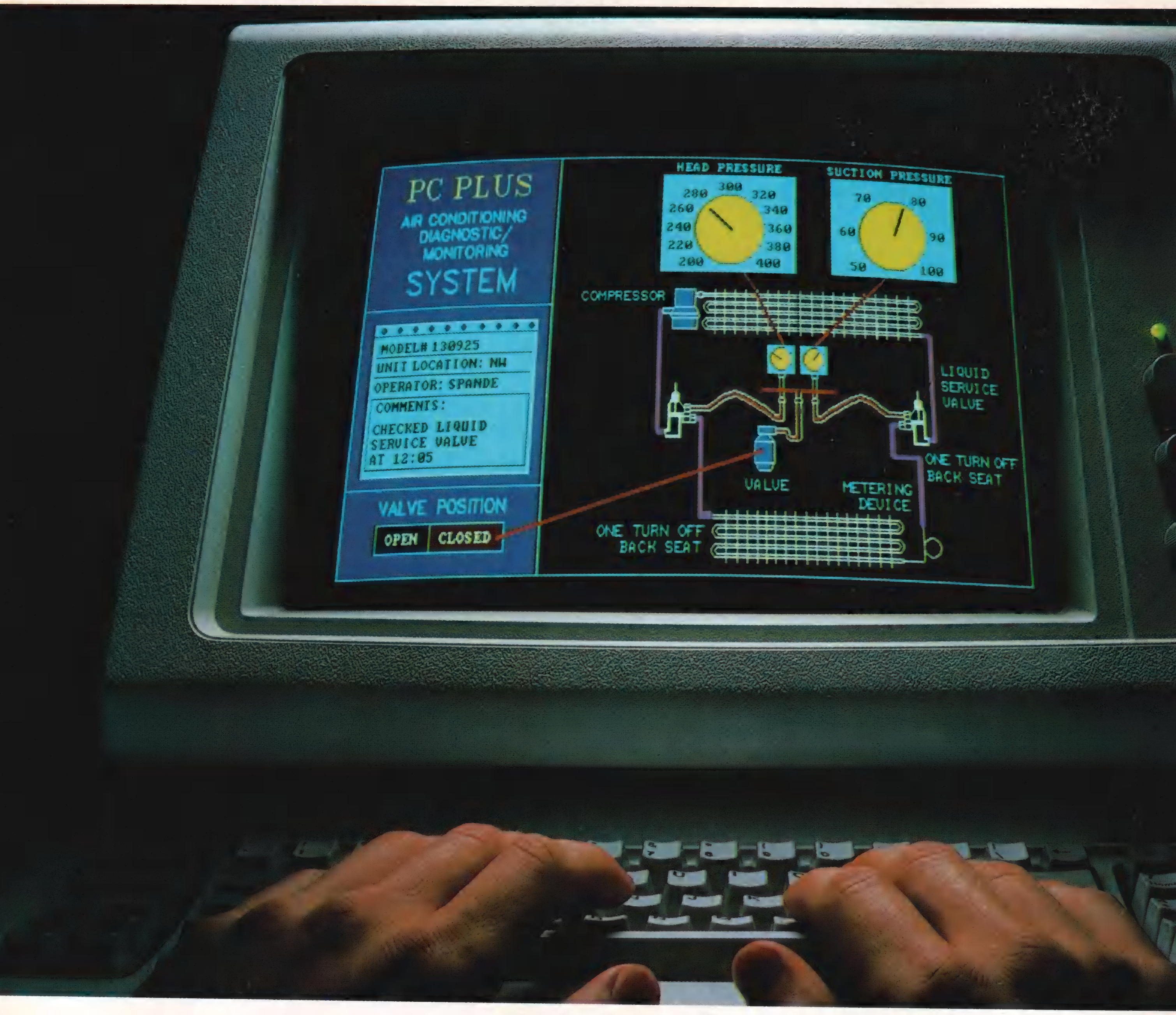
SantaRita

For the location of your nearest Santa Rita Software dealer, or to order direct, call us at 1-214-727-9217. We'd like to hear from you.

Santa Rita Software
1000 E. 14th Street, Suite 365
Plano, Texas 75074

The RESIDENT EXPERT System

Texas Instruments has system developers need.



MODEL# 130925
UNIT LOCATION: NW
OPERATOR: SPANDE
COMMENTS:
CHECKED LIQUID SERVICE VALVE AT 12:05

"Personal Consultant™ Plus . . . offers a very fine expert system development and delivery tool that already has a proven record with end-users."

— *Susan Shepard, AI Expert*

Personal Consultant Plus 3.0 Standard Features
- Frames, rules, meta rules and procedures
- Forward/backward chaining
- Confidence factors
- Regression testing and rule tracing
- End-user explanation facilities
- Graphics image capture and display
- Interfaces to dBase™, Lotus 1-2-3™, DOS files, .EXE or .COM programs, "C"
- Complete LISP development environment
- 2-megabyte expanded/extended memory support
- Mouse support
- Context sensitive help
- "Getting Started" tutorial-style manual

Personal Consultant Images
- Optional add-on package to PC Plus (3.0)
- Allows integration of "active images" into

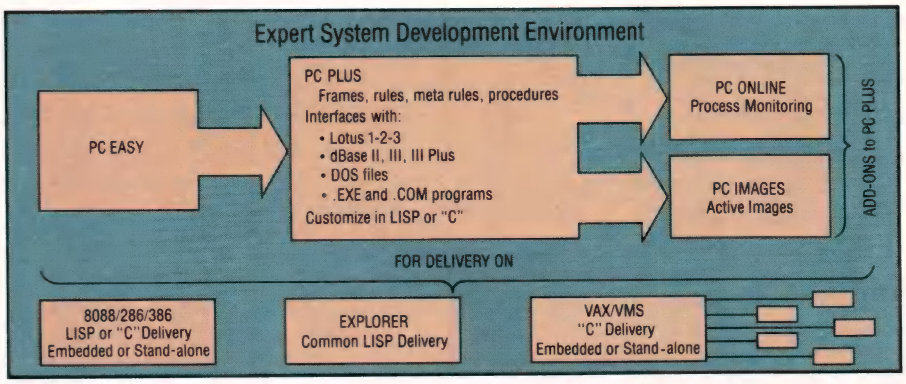
what serious expert Power tools.

knowledge bases
- Interactive dials, gauges, forms and selection images
- Multiple images can be combined on same screen
- "Getting Started" tutorial-style manual

Personal Consultant Online
- Optional add-on package for PC Plus (3.0)
- ONLINE expert systems that interact directly with process data
- Multiple interfaces to data acquisition and analysis programs
- Knowledge base synchronization with process data
- Functions for historical and predicted trends
- Special user interface/reporting capabilities
- "Getting Started" tutorial-style manual

Among all the expert system development tools available for personal computers today, none deliver the power and flexibility of TI's Personal Consultant series.

Personal Consultant Easy is ideal for getting started, and is upwardly compatible with the higher functionality of PC Plus. For experienced developers, Personal Consultant Plus and its optional add-on enhancements, Online and Images, were designed to help solve a broader range of complex problems.



Personal Consultant Plus. Full power for an affordable price.
At $2,950, PC Plus has proven to be one of the richest and most flexible problem-solving tools available for the development of complex knowledge-based systems. Designed to take advantage of today's more powerful 286/386 DOS-based computers, or TI's Explorer™ Symbolic Processing System, the new 3.0 version of PC Plus provides powerful standard features and a continuing growth path with the addition of either PC Images or PC Online, or both.

Personal Consultant Images. Picture an expert system with interactive graphics.
At $495, PC Images enables developers to create knowledge-based applications that incorporate complex graphical "active images." User-interactive dials, gauges, forms and selection images provide a more exciting visual data input and output style.

Personal Consultant Online. The expert system as part of the process.
At $995, PC Online allows the developer to design expert systems which interact directly with process data, as opposed to input from a human operator. Designed for intelligent process monitoring applications, this optional package helps deliver expertise that is "online all the time."

Application delivery as flexible as the tools themselves.
Delivery can be in LISP for flexibility, or "C"* for maximum speed and portability. Our "C" options support either stand-alone or "embedded" knowledge bases. Options are available for DOS-based PCs, TI's Explorer, and DEC's VAX™ line of multi-user minis running under VMS™.

"Texas Instruments has done more than any other company to educate people about AI, to popularize it, and to make useful AI tools available at reasonable prices."
— Jim Seymour, PC Magazine.

Technical support, training courses and Knowledge Engineering Services are available for the Personal Consultant products. If you have a question about any of our expert system power tools, we have the answer.

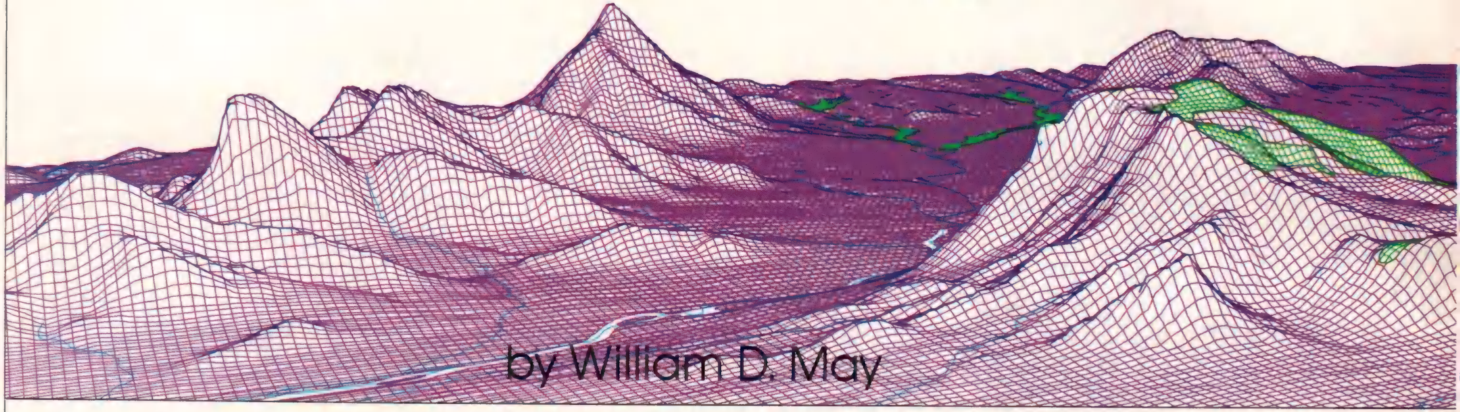
Pick up the phone and gain a powerful advantage.
Call **1-800-527-3500** for technical overviews of our products and a PC Plus case histories brochure which details how our power tools are being put to work today.

36106

*Available 4Q 1987.

TEXAS INSTRUMENTS

3-D Images from Contour Maps



This contour-analysis algorithm vividly demonstrates the gulf between human cognition and an algorithmic procedure.

Contour maps have intrigued me ever since my introduction to them in high-school geology. By studying a contour map, I could learn about inaccessible parts of the world and discover unexpected possibilities in my backyard. Yet for all its wealth of information and detail, a contour map is less than intuitive. You must say to yourself, "these contours are very close together, so this area must be very steep," or "these contours are widely spaced, so this area is fairly level."

It would be wonderful to be able to feed a contour map into a machine that could project images of the terrain in three dimensions. Then you could rotate the view, zoom in on interesting spots, change to different viewpoints, and generally get an extremely tangible feeling for the characteristics of the landscape.

This article is about a set of algorithms that takes a step toward achieving these objectives. The algorithms analyze a binary image of a contour map (such as that produced by an image scanner) and produce a representation of the image that can be used by a hidden-line algorithm. The hidden-line algorithm can then produce 3-D views from any mathematically valid viewpoint. Figures 1 and 2, page 20, show the before and after of this transformation.

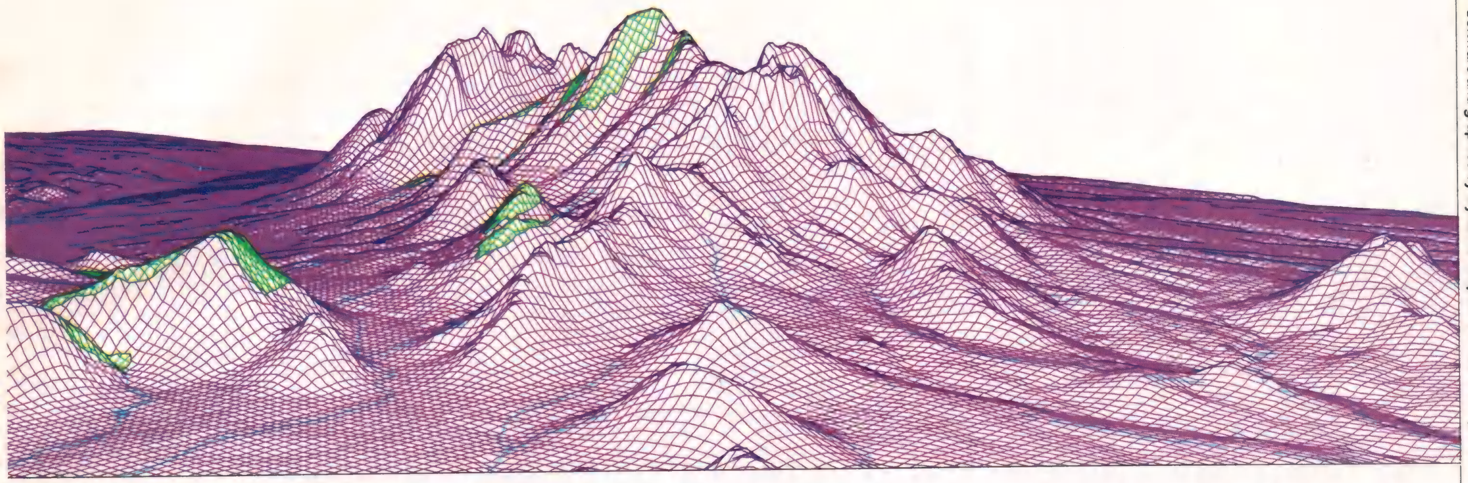
In many respects the means of transforming the contour map into three dimensions is as interesting as the transformation itself. Seeing the contours on a contour map is trivial for the human eye and brain but complex for a computer. A large part of the problem stems from bandwidth: the eye/brain seems to see an entire image, its components, and the relationships among its components simultaneously, but the computer "sees" an image one pixel at a time. The low bandwidth of the computer's vision makes it extremely difficult for the computer (that is, its programmer) to identify objects and to perceive their relationships.

The approach I have taken uses some geometry to overcome this lack of cognitive ability, first to find and identify the contours in the image and then to analyze the contour relationships (nesting). The drawback of using a geometric solution (as opposed to an "intelligent" or "learning" solution) is that this approach is useful only for this particular application. But I think this drawback is more than offset by the ability to do something that the eye/brain cannot: use the analysis to build a 3-D image from a 2-D map.

Of course, contour representation is used in areas other than geography, such as medicine, chemistry, and astronomy. The contour-analysis algorithm I have developed can be used in these situations as well, although the interpretation of the resulting "image" will be quite different.

I should caution you that the current state of this algorithm falls somewhat short of my original objective, the major problem being with the input of the map. The algorithm needs an image that consists of topologically correct contours. This means that continuous, closed contours on the map must in fact be continuous and closed in the image and that contours don't cross or merge.

William D. May, 20A-M23, Arthur D. Little Inc., Acorn Pk., Cambridge, MA 02140. Bill is a consultant specializing in system development.



Software and graphics by Dynamic Graphics, Inc., Berkeley, CA.

In real life, such as using a U.S. Geological Survey map, this requires that extraneous information be filtered out of the image and that the contours themselves get scanned accurately. Both requirements are difficult to achieve. U.S. Geological Survey maps contain a wealth of symbols, lines, markings, grids, shadings, and so on that interfere with the filtering process. The requirement for continuous lines is also difficult to achieve. First, the maps often place elevation information directly on the contour line, which is easy for humans to read but difficult for computers to interpret. Second, scanners have difficulty accurately placing the very fine lines used for contours, so the result sometimes looks more like a fog of dots where the contour is supposed to be than a continuous line.

From the Map to Three Dimensions

A program that transforms contour maps into 3-D views requires several steps to achieve its objective. I have summarized these steps below. The process draws on diverse subjects, such as image processing and 3-D graphics. The focus of this article is mainly on the algorithms for analyzing contour map information and building a 3-D representation from the analysis. These tasks correspond to steps 3 through 5 below.

1. Getting the map image into the computer. I use images stored in Macintosh MacPaint format, which can be created using MacPaint or via scanners such as Thunderscan. The MacPaint format is simple, consisting of 512 bytes of header information (which is usually ignored) and a bit map in which each bit represents a single black or white pixel in the image. Apple's Macintosh Tech Note #86 explains the MacPaint format in detail and provides sample Pascal code for reading and writing MacPaint documents.
2. Cleaning up the image. Extraneous image data —that is, data that is not part of a contour—must be removed before trying to analyze the map. In many cases some human work is needed to overcome imperfections in the scanning process.
3. Analyzing the contour image. This refers to finding all the contours in the bit map, "remembering" which are nested in which, and storing them in a way that can be used to build a 3-D image. The contour-analysis algorithm builds two data structures during the analysis. One is an array with one entry for each contour discovered in the map. Each array element contains the coordinates of one point on the contour; the index of the enclosing contour; and, eventually, the contour's elevation. The algorithm also stores every point that is on a contour as a node in a tree. The key for the node is the point's coordinates, and the datum consists of the array index of the contour the point is on. These two data structures make it easy to access the contours directly or to access contour information when given any point on the contour. This in turn makes the translation into three dimensions easy.
4. Assigning elevations to the contours. A restriction of the contour-analysis algorithm is that it cannot read elevations directly from a digitized image. In some applications this does not matter; in others, such as U.S. Geological Survey maps, the ability would be quite useful. That would be a different and very large project, though, and leaving it out, as I have done here, does not affect the basic logic of the algorithm. In lieu of reading the elevations from the map, I have implemented the simple case in which all nested regions are assumed to be ascending. Given the base elevation of the map, the program simply increments the elevation for each level of nesting in the image. The more general solution would allow users to override elevation assignments manually.
5. Creating an internal 3-D representation of the terrain.

(continued from page 19)

After the contour analysis, the program builds a rectangular grid as the transformed representation of the contour map. The grid is simply an array in which each cell contains the elevation at a point on the map. The larger the grid, the more finely it covers the map and the greater the detail in the resulting image.

6. Displaying a projection of the grid. The final step is to use a hidden-line algorithm to display the grid as a 3-D projection on the screen. Hidden-line algorithms allow the viewer to observe the scene from any angle in a fairly realistic format.

How the Contour Analysis Works

Two problems must be solved in order to analyze a contour map. The algorithm must find all the contours, and it must determine their nesting relationships.

The contour search begins by using a 2-D search of the image—that is, starting at the top left and scanning across each row until it finally reaches the bottom right. Each time it finds a contour (a black pixel), it determines if it is a contour that was previously "seen." If not—that is, if the program has found a new contour—it adds the contour to its internal data structures. In either case the search continues.

Unfortunately, this search pattern is too simple to fully serve your needs. The problem is that it provides no easy way to track how contours are nested. In order to track the nesting relationships you must specify a contour and have the contour-analysis algorithm find a single level of contours within it, as shown in Figure 3, below. The interiors of the contours within the region should not be searched (yet). This procedure makes the tracking of nesting easy: the enclosing contour is the "parent" of the contours discovered within it. Assuming you have this ability, you then use the top-left to bottom-right search to find the lowest level of contours in the image and use your new search algorithm to

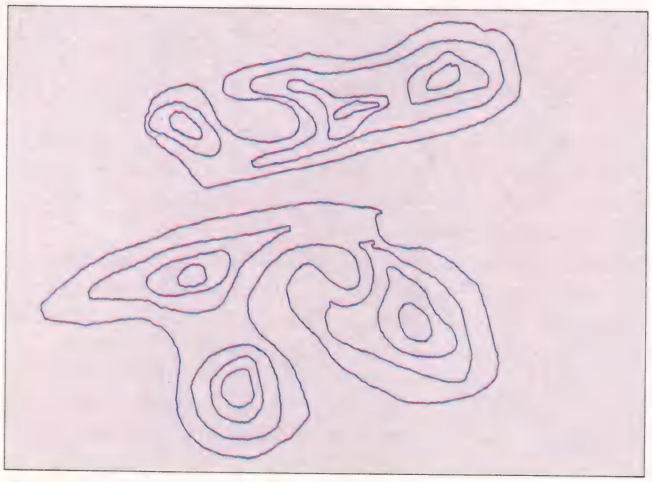


Figure 1: *A sample contour map*

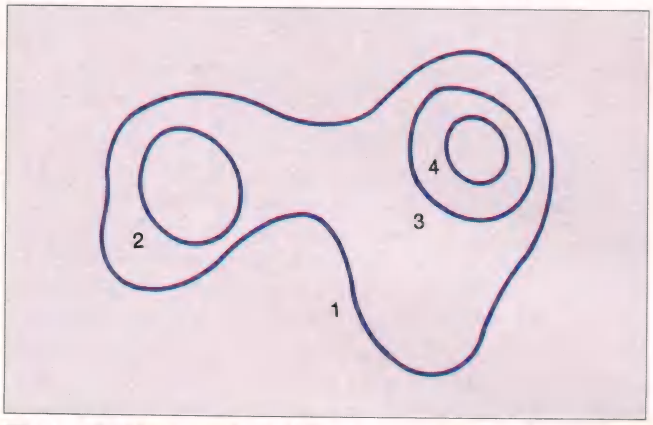


Figure 3: *The search within contour 1 will find contours 2 and 3 but not 4. Contour 4 will be found by a search of contour 3.*

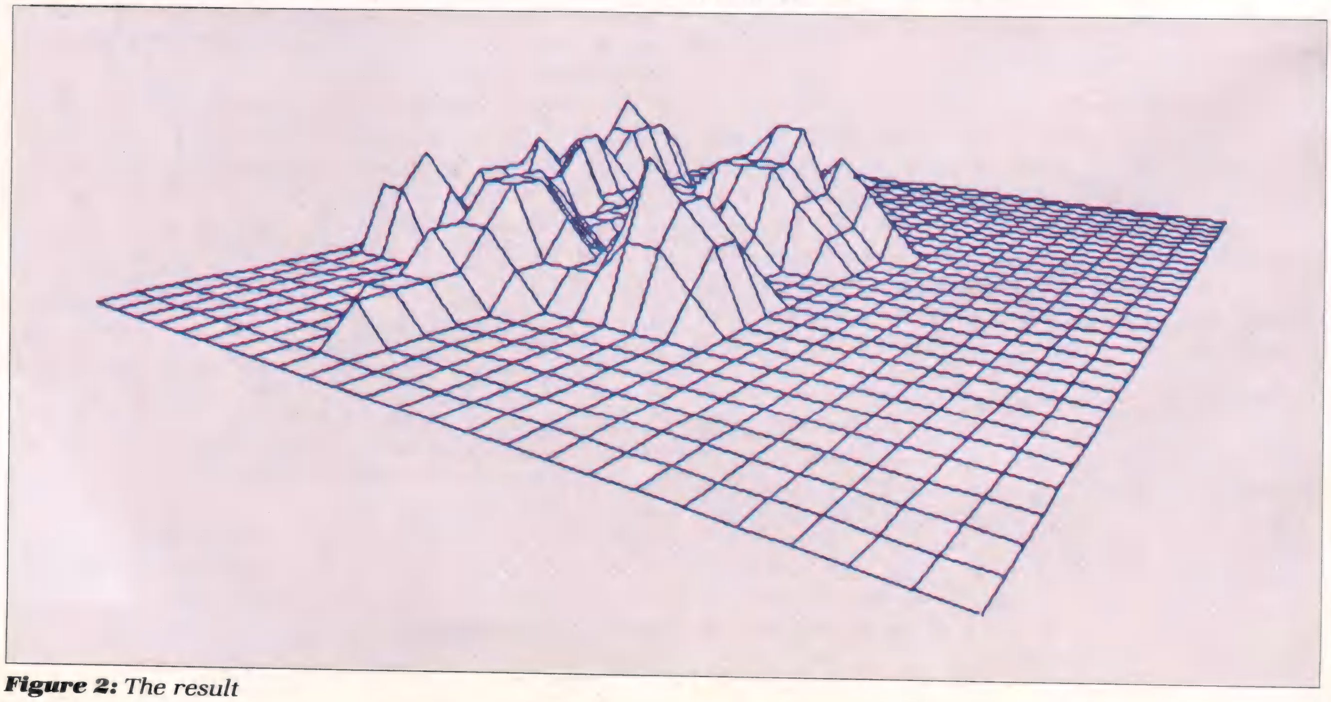


Figure 2: *The result*

Is your PC an endangered species?

With PC technology evolving at breakneck speed, it's become survival of the fastest. But before you decide your PC, XT or even AT is doomed to extinction, take a look at PC Technologies' complete family of enhancement boards:

386 Express™ — Our 16-MHz 80386 accelerator — specially suitable for LANs — will make your IBM AT two to three times more productive *and* maintain full software compatibility including OS/2, all at a remarkably low price — $995.

286 Express™ — The first half-slot 80286 accelerator card and winner of AFIPS/FORTUNE Magazine's Hardware Product-of-the-Year Merit Award. Engineered to deliver two to six times more speed for your PC, XT or compatibles.

286 RAMracer™ — Two performance upgrades, one low price. Combines 8-MHz 80286 acceleration *and* up to 2MB of EEMS/EMS- compatible expanded memory for your larger, demanding software applications.

RAMpartner™ — Batter down the DOS 640K memory barrier with up to 2MB of expanded memory. Ideal for spreadsheets, databases, windowing software . . . and as a companion to the RAMracer or AST and Intel expanded memory products.

286 Rainbow Plus™ — Our one-slot, five-function board contains an Enhanced Graphics Adapter, a 10-MHz 80286 accelerator, *PLUS* a Microsoft® InPort™ mouse interface, parallel printer port, and clock/calendar.

To find out how easy — and economical — it is to take your PC off the endangered species list, see your nearest PC Technologies dealer. Or call us today, direct, at 800-821-3086 (outside Michigan) or 313-996-9690.

PCT PC Technologies Inc.

704 Airport Blvd., Ann Arbor, MI 48108
313-996-9690/Telex 503589/FAX 313-996-0082,
800-821-3086 (outside Michigan)

CIRCLE 323 ON READER SERVICE CARD

(continued from page 20)

search the interior of each of the newly discovered contours (and then use it again to search the interior of any contours within them, and so on . . .).

This new search hinges on being able to differentiate the interior or exterior of an enclosed region. This requires a simple observation: if the program is following a contour in a counterclockwise direction, and it is currently following a downward arc, then the interior lies to the right of the current position, as seen in Figure 4, below. Alternatively, if the program is following a contour in a clockwise direction, and it is currently on a downward arc, then the exterior lies to the right of the current point.

Using this observation, the sequence of events to search the interior of a region is this: Trace around the bounding contour in a counterclockwise direction, and at each point on a downward arc, scan to the right (that is, the interior) until you find a new contour or find the other side of the bounding contour. On finding a new curve, traverse this curve in a clockwise direction, still scanning to the right on a downward arc, as indicated in Figure 5, below. Because the traversal is now reversed, this scan will now be the exterior of the new contour—that is, the interior of the original contour but on the opposite side of the nested contour. Applied to the bounding curve and all the interior curves, this procedure scans the interior of the bounding contour but not the interior of any contours nested within the bounding contour. For example, in Figure 3 a search of contour 1 done in this way will encounter contours 2 and 3 but not contour 4. Contour 4 will be discovered only when a search is made of the interior of contour 3. In fact, the entire search is reduced to this process by treating the border of the image itself as a large contour.

Following the contour analysis, the elevation assignments are made by stepping through an array of the contours found and assigning an elevation equal to the enclosing contour's elevation plus the elevation increment. This is where I use the assumption that all nested regions are rising.

Finally, an array of elevations is developed to be used as input to the hidden-line algorithm. The array elements are calculated by traversing the image along the grid lines. When a traversal begins, it sets its current elevation to the base elevation of the contour map, and each time it reaches a grid intersection it assigns the current elevation to the corresponding element of the array. The traversal also looks for intersections with contours on the map. These intersections signal a change in the current elevation. When it encounters a contour, the program first retrieves the elevation of the contour from the contour array described earlier.

At first glance it may seem that this new elevation should become the current elevation. It's not that simple. If the traversal is going into a nested region, then the new elevation is the elevation of the region it is now entering and thus should become the current elevation. If the traversal is leaving a nested region, however, the new elevation is the elevation of the region just left.

For example, imagine traversing the following simple map: the base of the map has a zero elevation, and there is a plateau in the center with an elevation of 100 feet. When the traversal begins, it uses the base elevation (0) as the current elevation. When it encounters the contour around the plateau, because it is entering a nested region, it assigns 100 to the current elevation. When it reaches the other side of the plateau, it once again meets the contour with an elevation of 100. Because you are now leaving a nested region, however, this signals that the elevation is no longer 100 feet, and, in this case, it should now be the base elevation of 0.

There is a data structure that naturally resolves this problem: the stack. Each time the program encounters a contour, it compares the contour's elevation to the top of the stack. If the elevation is different, it pushes the new elevation. If the new elevation is the same as the top of the stack, it pops the stack. In either case the current elevation is the number on the top of the stack.

Using the Program

Using the program is quite simple. The main program sets up a contour array (it must be larger than the expected number of contours) and a pointer to the tree of points as global variables. It then calls *find_all_contours( )*, passing a pointer to the bit map to be analyzed. On return it calls *make_hidlinpix( )* to assign elevations, which in turn calls *make_grid( )* to calculate the grid

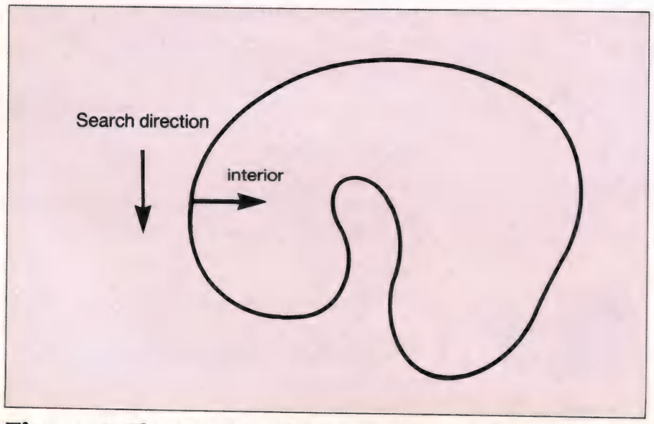


Figure 4: *The interior of the region is to be right when traversing a descending arc.*

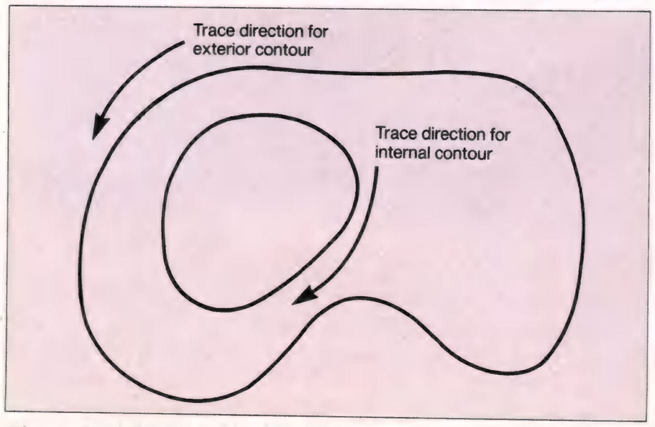
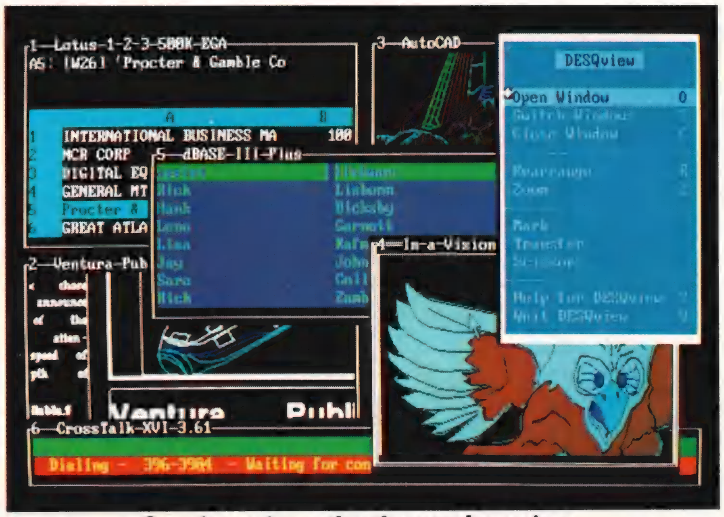

Trace direction for exterior contour
Trace direction for internal contour

Figure 5: *The trace for the external curve is counterclockwise; for internal curves it is clockwise.*

How to tell the difference between DESQview™ 2.0 and any other environment.



One picture is worth a thousand promises.

Selecting DESQview, the environment of choice, can give you the productivity and power you crave, without the loss of your old programs and hardware. If you like your existing programs, want to use them together, transfer data between them, print, sort, communicate with or process-in-background, yet still have the need to keep in place your favorite PC(8088, 8086, 80286 or 80386), DESQview is the "proven true" multitasking, multi-windowing environment for you. Best of all, DESQview 2.0 is here now, with all the money saving, time saving, and productivity features that others can only promise for the all-too-distant future.

And with DESQview's new graphics enhancements for Hercules, CGA, EGA, and VGA, Version 2.0 still offers the same award winning and pioneering features for programs that earned DESQview its leadership, only now you can also run desktop publishing programs, CAD programs, even GEM-™ Topview-™ and Microsoft Windows-™ specific programs. In some cases you'll add as little as 10–40K to your system overhead. Now you can have multi-tasking, multi-windowing, break the 640K habit too and still get an auto dialer, macros, menus for DOS and, for advanced users, a new complete application programmer's interface capability. No wonder that over the years, and especially in recent months, DESQview, and now DESQview 2.0 have earned extravagant praise from some of the most respected magazines in the industry.

INFO WORLD
DESQview 2.0
9.1
July 6, 1987

"Product of the Year" by readers vote in InfoWorld.

"Best PC Environment" by popular vote at Comdex Fall in PC Tech Journal's "System Builder" Contest.

"—I wouldn't want to run an IBM or compatible computer without DESQview"—InfoWorld, Michael Miller.

"A colossus among windowing environments"... "will run almost anything"—PC Week, Marvin Bryan.

"Windows, promises, but DESQview delivers"—MICROTIMES, Birell Walsh.

No other environment has consistently pioneered features, openness, and productivity. See for yourself. Send in the coupon. The possibilities are endless with DESQview 2.0.

Attention Programmers: For more information about Quarterdeck's API, and future 386 program extensions, call us today.

SYSTEM REQUIREMENTS
IBM Personal Computer and 100% compatibles (with 8086, 8088, 80286 or 80386 processors) with monochrome or color display; IBM Personal System/2 • Memory: 640K recommended; for DESQview itself 0-145K • Expanded Memory (Optional): expanded memory boards compatible with the Intel AboveBoard; enhanced expanded memory boards compatible with the AST RAMpage • Disk: Two diskette drives or one diskette drive and a hard disk • Graphics Card (Optional): Hercules, IBM Color/Graphics (CGA), IBM Enhanced Graphics (EGA), IBM Personal System/2 Advanced Graphics (VGA) • Mouse (Optional): Mouse Systems, Microsoft and compatibles • Modem for Auto-Dialer (Optional): Hayes or Compatible • Operating System: PC-DOS 2.0-3.3; MS-DOS2.0-3.2 • Software: Most PC-DOS and MS-DOS application programs; programs specific to TopView 1.1, GEM 1.1 and Microsoft Windows 1.03 • Media: DESQview 2.0 is available on either 5¼" or 3½" floppy diskettes

Rush me DESQview 2.0! Today! **DDJ 1187**

No. of Copies	Media 3½"/5¼"	Product	Retail Price ea.	Total
		DESQview 2.0	$129.95	$
		Shipping & Handling USA / Outside USA	$ 5.00 / $ 10.00	$
		Sales Tax (CA residents)	6.5%	$
			Amount Enclosed	$

Payment: ☐ Visa ☐ MC ☐ AMEX ☐ Check

Credit Card: Valid Since ______/______ Expiration ______/______

Card Number: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Credit Card Name ______

Shipping Address ______

City ______ State ______ Zip ______ Telephone ______

Mail to: Quarterdeck Office Systems, 150 Pico Boulevard, Santa Monica, CA 90405.
NOTE: If you own DESQview call us for a special upgrade offer, or send in your DESQview registration card. AST Special Edition users included.

DESQ view™ Quarterdeck

See us at COMDEX/Fall '87
November 2–6, 1987
Las Vegas Hilton Hotel
Las Vegas, Nevada

Quarterdeck Office Systems • 150 Pico Boulevard, Santa Monica, CA 90405 • (213) 392-9851

CIRCLE 284 ON READER SERVICE CARD

(continued from page 22)

elevations. The current implementation converts the grid to a disk file that is then used as input to a hidden-line algorithm.

The previous quick overview of the contour-analysis and grid-construction algorithms glosses over most of the gory details, such as coordinating progress and tracing curves. I'll now describe these in more detail.

Find_all_contours( )

Find_all_contours( ), as its name implies, is responsible for coordinating the entire contour search. The progress of the contour-analysis algorithm is based on a queue (a first-in/first-out data structure). The queue is *find_all_contours( )*'s means of communicating with the lower-level function *find_contours( )*. *Find_all_contours( )* calls a function to enqueue the points on a contour of interest and then calls *find_contours( )*. *Find_contours( )* then dequeues all points, and those on a descending arc are used as the starting points for rightward scans.

The queue is built by tracing contours in a counterclockwise direction and includes the location of each point on the contour as well as a chain code for the point. The chain code is an integer from 0 to 7 that indicates the direction moved from the previous contour point to the current contour point, as shown in Figure 6, below. Thus a chain code of 0 indicates a movement to the right, 2 is a movement up, 4 is to the left, and 6 is down. *Find_all_contours( )* initiates the entire contour analysis by enqueuing the borders of the image—that is, by treating the border of the image as a contour itself. Because only the left-hand side of the image will have chain codes in the range 5 to 7, only the left side really needs to be enqueued. *Find_all_contours( )* then calls *find_contours( )*, which in turn uses the queue to do a single-level contour search. Any contours found by *find_contours( )* are placed in the contours array.

On return from *find_contours( )*, there will usually be some contours stored in the contour array. *Find_all_contours( )* then steps through each unsearched contour in the array. At each iteration, *find_all_contours( )* calls a function to enqueue (by tracing) the points on the contour and once again calls *find_contours( )* to search the interior of the new region. When the interiors of all contours in the array have been scanned, the search is complete.

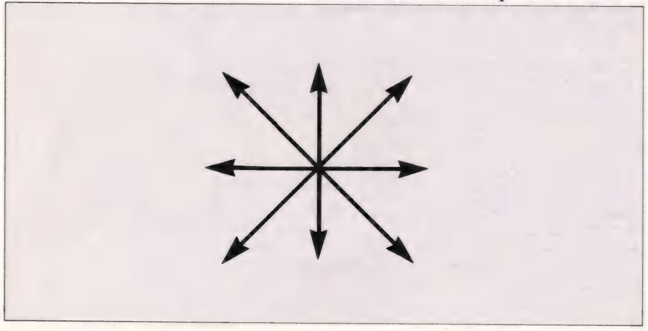


Figure 6: *Chain codes*

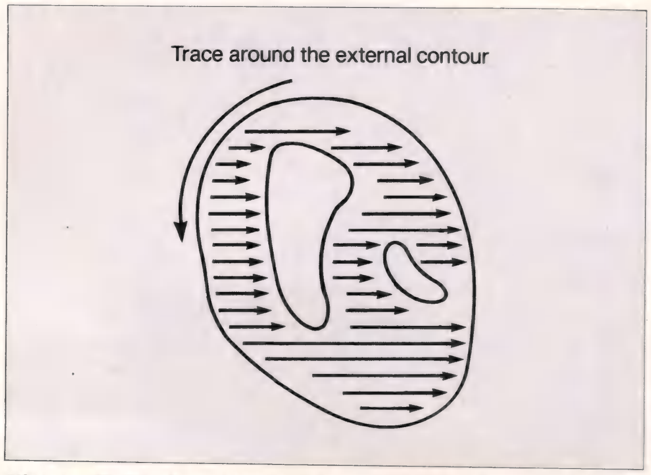


Figure 7: Find_contours( ) *searches the interor of a given contour for nested contours.*

Find_contours( )

Find_contours( ) is responsible for finding a single level of contours nested within a given region, which is that region enqueued by *find_all_contours( )* (see Figure 7, below). Because the queue was created by tracing the contour in a counterclockwise direction, the contour interior will be found by scanning to the right of points whose chain codes are in the range 5 to 7. For each chain code in this range, the function *search_x( )* is used to perform the actual scan of the bit map.

The scan can return either of two results: it can encounter a known contour (which can be the other side of the enclosing contour, the edge of the bit map, or an internal but previously discovered contour), or it can find a new contour. If it finds a new contour, then *find_contours( )* calls *trace( )* to traverse the new contour in a clockwise direction. Once again the trace enqueues the points on the contour. When *find_contours( )* reaches these points in the queue, it will result in the scan starting on the right side of the new contour and still moving right. The result is that the interior of the enclosing contour is scanned, except for the interiors of any nested contours.

There are two problems with this general procedure: tabletops and peaks (see Figure 8, below, for examples). Tabletops are horizontal runs to the right. The problem is that the point on the downward arc will start the scan. The scan will immediately encounter a point on the contour (that is, part of the tabletop) and stop, thinking (incorrectly) that it has reached the other side. The solution to this problem is to look at the next chain code as well as the current one. If the current chain

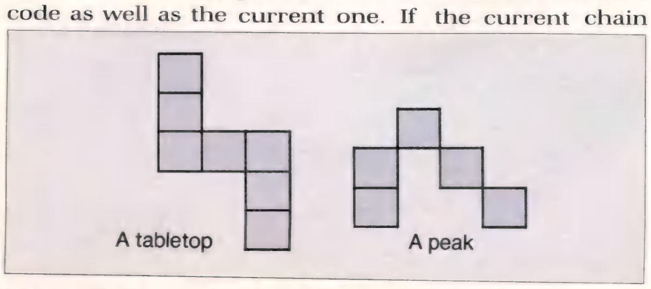


Figure 8: *Examples of a tabletop and a peak*

Clarify and document your source listing and get an "organization chart" of your program's structure

with two NEW utilities from Aldebaran Laboratories, for C, BASIC, Pascal, dBASE,® FORTRAN and Modula-2 programmers.

Now works with FORTRAN

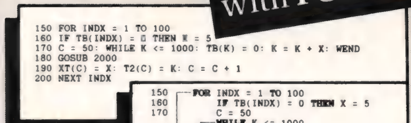
"Occasionally, a utility comes along that makes a programmer's life much easier. SOURCE PRINT is such a program. It contributes to the programmer's job by organizing code into a legible format and by helping to organize the documentation and debugging process."

— PC Magazine
Sept. 16, 1986

Source Print™

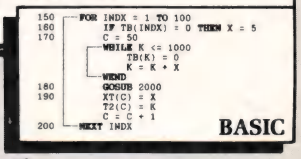
organizes your source code, simplifies debugging, and makes documentation a snap! It lists one or more source files with informative page headings and optional line numbers, while offering invaluable features:

The Index (Cross-Reference list) saves you time by showing exactly where variables are used and where functions, procedures, and routines are called. Locations where new values may be assigned to variables are shown, making it easy to track down that mysterious value change.



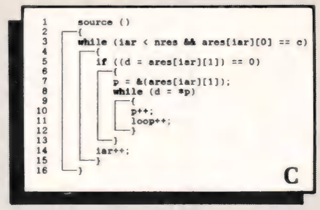
```
150 FOR INDX = 1 TO 100
160 IF TB(INDX) = 0 THEN X = 5
170 C = 50: WHILE K <= 1000: TB(K) = 0: K = K + X: WEND
180 GOSUB 2000
190 XT(C) = X: T2(C) = K: C = C + 1
200 NEXT INDX
```

Before

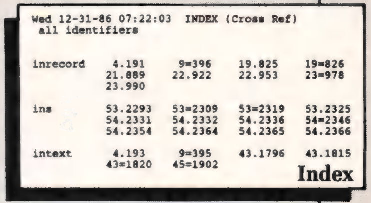


```
150  ┌──FOR INDX = 1 TO 100
160  │       IF TB(INDX) = 0 THEN X = 5
170  │       C = 50
     │  ┌────WHILE K <= 1000
     │  │         TB(K) = 0
     │  │         K = K + X
     │  └────WEND
180  │       GOSUB 2000
190  │       XT(C) = X
     │       T2(C) = K
     │       C = C + 1
200  └──NEXT INDX
```

After

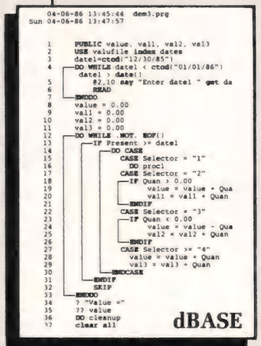


```
      source ()
  ┌───{
  │   while (iar < nres && ares[iar][0] == c)
  │  ┌───{
  │  │   if ((d = ares[iar][1]) == 0)
  │  │  ┌───{
  │  │  │   p = &(ares[iar][1]);
  │  │  │   while (d = *p)
  │  │  │  ┌───{
  │  │  │  │   p++;
  │  │  │  │   loop++;
  │  │  │  └───}
  │  │  └───}
  │  │   iar++;
  │  └───}
  └───}
```



```
Wed 12-31-86 07:22:03  INDEX (Cross Ref)
 all identifiers

inrecord      4.191      9=396     19.825     19=826
             21.889     22.922     22.953     23=978
             23.990

ins         53.2293    53=2309    53=2319    53.2325
            54.2331    54.2332    54.2336    54=2346
            54.2354    54.2364    54.2365    54.2366

intext        4.193      9=395    43.1796    43.1815
            43=1820    45=1902
```

$97.00



```
    04-06-86 13:45:44  dem3.prg
Sun 04-06-86 13:47:57

      PUBLIC value, val1, val2, val3
      USE valufile index dates
      datel=ctod("12/30/85")
   ┌──DO WHILE datel < ctod("01/01/86")
   │   datel > date()
   │      @2,10 say "Enter datel " get da
   │      READ
   └──ENDDO
      value = 0.00
      val1 = 0.00
      val2 = 0.00
      val3 = 0.00
   ┌──DO WHILE .NOT. EOF()
   │  ┌──IF Present >= datel
   │  │  ┌──DO CASE
   │  │  │    CASE Selector = "1"
   │  │  │      DO proc1
   │  │  │    CASE Selector = "2"
   │  │  │   ┌──IF Quan > 0.00
   │  │  │   │       value = value + Qua
   │  │  │   │       val1 = val1 + Quan
   │  │  │   └──ENDIF
   │  │  │    CASE Selector = "3"
   │  │  │   ┌──IF Quan < 0.00
   │  │  │   │       value = value - Qua
   │  │  │   │       val2 = val2 + Quan
   │  │  │   └──ENDIF
   │  │  │    CASE Selector >= "4"
   │  │  │      value = value + Quan
   │  │  │      val3 = val3 + Quan
   │  │  └──ENDCASE
   │  └──ENDIF
   │      SKIP
   └──ENDDO
      ? "Value ="
      ?? value
      DO cleanup
      clear all
```

Structure Outlining solves the problem of hard-to-see nested control structures by automatically drawing lines around them.

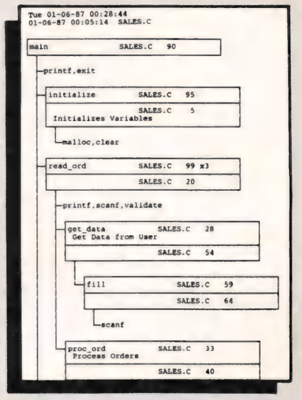
Automatic Indentation of source code and listings reduces your editing time and ensures indentation accuracy.

Plus . . . Source Print generates a table of contents listing functions and procedures. Keywords can be printed in boldface on most printers. Multi-statement BASIC lines can be split for readability. Functions and procedures can be drawn by name from one or more source files to form a new file.

Tree Diagrammer™

shows your program's overall organization at a glance. Ordinary program listings merely display functions, procedures, and subroutines sequentially, but do not display the relationships between these routines. Our revolutionary new Tree Diagrammer automatically creates an "organization chart" of your program showing the hierarchy of calls to functions, procedures, and subroutines. Recursive calls are indicated and designated comments in the source code will appear on the chart.

Tree Diagrammer helps you organize your program more logically. And you'll be amazed at how easy it is to debug when you see how your routines interact.



```
Tue 01-06-87 00:28:44
01-06-87 00:05:14  SALES.C

main                    SALES.C   90
 │
 ├─printf,exit
 │
 ├─initialize           SALES.C   95
 │                      SALES.C    5
 │   Initializes Variables
 │   └─malloc,clear
 │
 ├─read_ord             SALES.C   99 x3
 │                      SALES.C   20
 │   ├─printf,scanf,validate
 │   │
 │   ├─get_data             SALES.C   28
 │   │   Get Data from User
 │   │                      SALES.C   54
 │   │   └─fill                 SALES.C   59
 │   │                          SALES.C   64
 │   │       └─scanf
 │   │
 │   ├─proc_ord             SALES.C   33
 │   │   Process Orders
 │   │                      SALES.C   40
```

$77.00

Source Print and Tree Diagrammer both have easy-to-use menus with point-and-shoot file selection, and let you search for files containing a given string. For IBM PC and compatibles with 256K.

Join thousands of programmers who are working more efficiently using Source Print and Tree Diagrammer. Order these indispensable tools today. We ship immediately, and there's no risk with our 60-day money-back guarantee. **Order both and save. Only $155.00.**

800-257-5773 Dept. 58
In California:
800-257-5774 Dept. 58

MasterCard, VISA, American Express, COD. Add $5 for shipping/handling.

or see your local dealer!

Source Print and Tree Diagrammer are trademarks of Aldebaran Labs. dBASE is a trademark of Ashton Tate. Prices subject to change without notice.

Aldebaran Laboratories 3339 Vincent Rd. Pleasant Hill, CA 94523 415-930-8966

YES! Rush me ☐ Source Print @ $97. ________ ☐ Tree Diagrammer @ $77. ________
☐ Both $155. Ship/Handling $5. For CA add 6% tax ________ Total ________
Name ________
Company ________
Address ________
City ________ State ________ Zip ________
☐ Check enclosed ☐ VISA ☐ MasterCard ☐ American Express
Card # ________ Exp. Date ________
Signature ________ Phone # ________ 58

CIRCLE 350 ON READER SERVICE CARD

(continued from page 24)

code was 0 and the next chain code is 5–7, then the program has just reached the end of a tabletop and it should start to do a scan.

A peak is a single pixel at the top of a contour. Peaks present another problem: they are never on a downward arc and will never initiate a rightward scan. A peak can result in at most a single scan line being skipped. Thus, in the worst case, the only contour that can be missed is a single line of pixels. Because this is a degenerate case, I did not feel it needed to be resolved.

Search_x( )

Search_x( ) scans to the right of a given point until it finds a black pixel or reaches the edge of the bit map, as shown in Figure 9, below. It then returns a value of 1 if the pixel is on a new contour or 2 if it is either on a known contour or at the edge of the bit map. A pixel has been seen before if it can be found in a tree containing the pixels lying on contours.

The interrogation of individual pixels on an off-screen, bit-map image can be done in several ways. The Macintosh Toolbox contains the functions *GetPixel( )* and *GetCPixel( )* (in the Macintosh II's Color QuickDraw). However, I have used an assembly-language function, called *getpixel( )*, that accomplishes the same task by accessing the bit map directly. This *getpixel( )* is quite fast, but it lacks the flexibility and generality of its Toolbox counterparts. Any machine that supports bit-

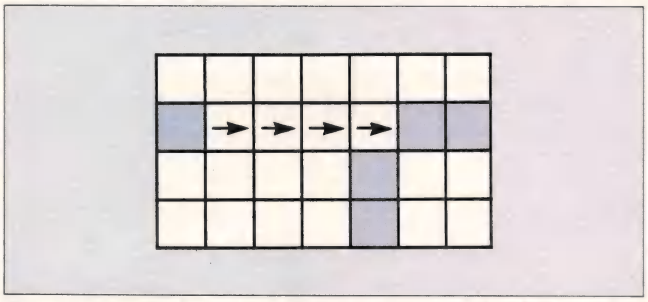


Figure 9: Search_x( ) *starts at a given pixel and searches to the right until it hits a contour or the edge of the image.*

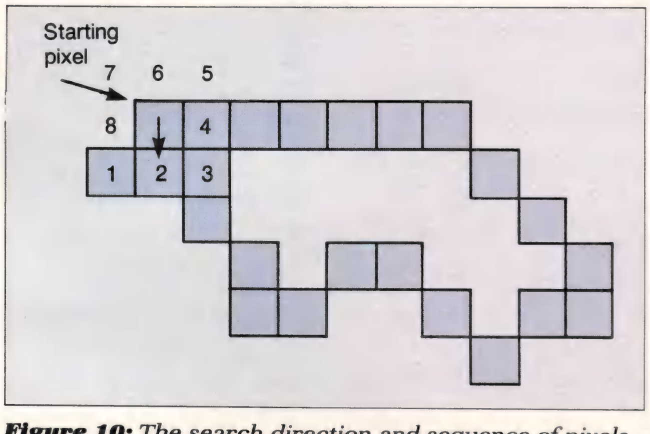


Figure 10: *The search direction and sequence of pixels searched during a contour trace ensures that the trace always follows the outermost border of the contour.*

mapped graphics will have counterparts to *getpixel( )* or *GetPixel( )* (the Microsoft C, Version 5.0, library now has a *_getpixel( )* function).

Trace( )

Trace( ) is the real workhorse of the contour-analysis algorithm. It traces a given curve in a clockwise or counterclockwise direction, and it simultaneously updates both the queue and a tree of points lying on the contour.

In order to follow a contour, *trace( )* uses the chain codes described earlier. An example can help explain the process. Assume that the program will traverse a contour counterclockwise. The calling function passes a point on the exterior of the contour to *trace( )*. First, the trace sets the initial search direction to 6 (down). It then looks at pixels with chain codes 5, 6, and 7 from the starting point—that is, "search direction–1," "search direction," "search direction + 1," all modulo 8 (see Figure 10, below). Note that the search starts the furthest to the exterior as is possible (a pixel at chain code 4 would mean that the starting pixel is not on the exterior).

The first hit then becomes the new search direction. If there is no hit at all, trace adds 2 to the search direction (modulo 8) and tries again. This process continues for three iterations at most, to prevent an infinite loop if the contour consists of a single point.

Note that if there is a break in the contour, this algorithm will circle around the end point and return back to the starting point. Also note that *trace( )* always proceeds around the exterior of a contour. If the contour is more than one pixel thick, *trace( )* will ignore the interior pixels. These points will be found later while scanning the interior of the region.

Building the 3-D Image

As indicated in the overview, the contour analysis itself is only the first step in producing a projection of the contour map. Once the analysis is complete, it must be converted into a form that a hidden-line algorithm can use as input. For purposes of demonstrating the contour analysis, I have used an algorithm published by L. Ammeraal in *Programming Principles in Computer Graphics*. This requires two further steps after the analysis itself.

First, the data from the analysis must be used to construct a 2-D array, with the value of each element of the array being the elevation at that point. This is done by the function *make_grid( )*, as described in the overview.

Next, the array itself must be converted into a file that can be used as input to the hidden-line algorithm. This is done by the function *make_hidlinpix( )*. A small section of this output file is shown in Table 1, page 28. It starts with the coordinates of the midpoint of the object to be drawn. Then follows a section that lists all the points in the array and their spatial coordinates (x, y, z). Thus point 1 is at spatial coordinates (0.0, 0.0, 0.0), 33 is at (0.0, 1.0, 0.0), and so on.

The key to the hidden-line algorithm is the following section, beginning with the word *Faces:*. This is a decomposition of the grid into triangular faces. Trian-

Smile if you're using the 'industry-standard' database.

The industry-standard was a great place to start, but it's kind of a sheep in wolf's clothing when it comes to serious business applications.

But Clipper, the leading-edge database system, is a tiger.

With Clipper, your present database applications run up to ten times faster because Clipper supports your existing dBASE programs, files and indexes.

And Clipper makes database networking a breeze, allowing multiple users to view, edit and add records in a shared file.

Then once you've solved your current problems, you can use Clipper's extended power to deal with your future needs.

Because Clipper is an extended database compiler/language that treats dBASE as a subset and removes the limits you're working with now.

Clipper includes dBASE-like commands for quick menus, fast screens and extended functions that make it easy to create user-friendly applications that don't look or act like dBASE. It handles arrays, more fields and memory variables and includes scores of enhancements that get you through your applications backlog sooner. And if there's something we've overlooked, you can create your own functions written in Clipper or add them in C and assembler.

Then once your programs are compiled, you can distribute as many copies as you like with full source code security and no LAN packs, no royalties, no licensing fees.

To get Clipper working for you, call (213) 390-7923 today.

We're not pulling the wool over your eyes.

Clipper™ is all business.

Nantucket®

CIRCLE 220 ON READER SERVICE CARD

(continued from page 26)

gles are used because all points on a triangle in 3-D space are located on the same plane (see Ammeraal's book for further explanation). Each face is defined by its three corner points listed in counterclockwise order: for example, the file shown here indicates that points 1, 34, and 2 form a triangular face. The points 1, 2, 33, and 34 form the corners of a rectangle. The program has broken this rectangle into two triangular faces: 1, 34, and 2; and 1, 33, and 34. The minus sign for point 34 indicates that the line segment from point 1 to point 34 should not actually be drawn, in this case because it forms a diagonal across a rectangle.

In fact, the grid faces are actually defined twice in the file: once as the top side (the normal viewpoint) and once as the under side; that's the reason for the apparent redundancy in the description of the faces. This technique allows you to view the grid both from above and below (underground if this were a topographic map!). This file is then read and processed by the program hidlinpix. The result is shown in Figure 2.

Portability and Optimization

The contour-analysis algorithm was written and tested on a 512K Macintosh and on a Macintosh II using Lightspeed C. In order to simplify the development and testing process, I modified a copy of the Lightspeed *stdio* library so that the console window opens to cover the lower quarter of the screen (instead of the entire desktop, as it normally does). The program then opens two windows: a window that shows the source image and a window that displays the progress of the contour analysis (a sort of debugging animation). The algorithms themselves are intended to be portable, but I have not actually tested the code in other environments, so there could be surprises in store.

I have used the Macintosh's graphics abilities to animate the code, so if you find the written explanation of the program complex, and if you are able to view the animation, you may find that it makes visual sense. There is some redundancy because of the need to trace each contour twice (once when first discovered, once again to search its interior), but the vast majority of points in the image are visited only once.

The code shown here uses one important optimization: the avoidance of the memory-management library supplied with Lightspeed C (that is, it avoids the use of *malloc( )*, *free( )*, and so on). Instead, I have used a nonstandard function called *getmem( )*. The rationale for this action is explained in detail in the source code to *getmem( )* itself. I was reluctant to include *getmem( )* with this article (it lacks its counterparts, such as *freemem( )*), but the speed improvement is quite dramatic; without *getmem( )* the program is slow, even on the Macintosh II.

```
           15.500000   12.500000    0.000000
  1         0.000000    0.000000    0.000000
  33        0.000000    1.000000    0.000000
  65        0.000000    2.000000    0.000000
  97        0.000000    3.000000    0.000000
  129       0.000000    4.000000    0.000000
  161       0.000000    5.000000    0.000000
  193       0.000000    6.000000    0.000000

  . . .

  Faces:

  1     -34    2#
  2      34   -1#
  1     -34   33#
  33     34   -1#
  33    -66   34#
  34     66  -33#

  . . .
```

Table 1: *A sample of the output file from the contour analysis*

Miscellaneous Functions

The contour analysis depends on several different data structures. I have avoided reinventing the wheel by using previously published code, particularly from some of Allen Holub's columns in *DDJ*. These borrowings include the queue and AVL tree code published in June 1985 and August 1986, respectively.

Conclusion

For the eye and brain, identifying objects and relationships in the visual world seems effortless. The contour map analysis algorithm is an example of the complexity of this process when performed by machine, even when restricted to a simple class of images. The algorithm vividly demonstrates the wide gulf between human cognition and an algorithmic procedure. In spite of the constraints placed on it, though, the machine is able to provide services that are impossible for the eye/brain; specifically, restoring a 3-D image from a 2-D representation.

Availability

All the source code for articles in this issue is available on a single disk. To order, send $14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call (415) 366-3600, ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

Bibliography

Ammeraal, L. *Programming Principles in Computer Graphics*. Chichester, England: Wiley, 1986.

Apple Computer. Macintosh Technical Support. *Macintosh Tech Notes*. Cupertino, Calif.: Apple Computer, 1983–1987.

Holub, Allen. Various C Chest columns. *DDJ*. 1986–1987.

Pavlidis, Theo. *Algorithms for Graphics and Image Processing*. Rockville, Md.: Computer Science Press, 1982.

DDJ

(Listings begin on page 66.)

Prospero Software

LANGUAGES FOR MICROCOMPUTER PROFESSIONALS

PRO FORTRAN-66
PRO PASCAL
PRO FORTRAN-77
CP/M micros
Amstrad CPC 6128
Amstrad PCW 8256
Any CP/M micro
Z80
Z80A
Z80B
68000 micros
Atari ST
Sinclair QL
Any PC
68008
68010
68020
MS-DOS micros
IBM PC/XT/AT
Amstrad PC1512
Compaq Deskpro
Olivetti M24
Zenith PC
Apricot Xen
RM Nimbus
Any PC
8086
8087
80186
80286
80287
80386
80387

Prospero Software is dedicated to languages and to customer support. For an opinion ask a colleague; for information call 01-741 8531 or write to Prospero Software Ltd, 190 Castelnau, London SW13 9DH, England.

CIRCLE 160 ON READER SERVICE CARD

A Graphics Toolbox for Turbo C

by Kent Porter

The first article in a two-part series on designing a library for PC graphics

That Borland's Turbo C comes equipped with 350 functions and macros is pretty impressive, until you realize that not one of them has anything to do with graphics. To fill this void I've constructed my own version of a graphics toolkit for Turbo C.

To make the information more easily digestible, I've divided the project into two bite-size articles. This piece introduces a basic library of graphics calls and shows how to put them to work in pixel-oriented graphics; Part 2 (to appear in December's *DDJ*) will combine this graphics library with data structures and C functions to control pop-down menus, dialog boxes, and other appearance features that users have come to expect in contemporary software.

The MS-DOS Underpinning

IBM PCs, or equivalents, rely on ROM BIOS interrupt *10h* (16 decimal) for their graphics capabilities. This interrupt furnishes "generalized" functions for various aspects of text and APA (all-points addressable, or pixel-oriented) graphics. I say "generalized" because the ROM BIOS accommodates numerous modes, some of which may not be available on certain hardware configurations. For example, the ROM BIOS handles video modes for the EGA monitor and the PCjr that don't exist when a CGA or monochrome adaptor is present in the system.

There are 16 core functions in the ROM BIOS video services. These functions govern display aspects such as the position or shape of the cursor, the video mode (text and APA options), the active display page, character output, pixel graphics, and others.

The ROM BIOS functions are not known for their speed. In general, they're adequate in text operations but less than adequate in APA graphics. You could devise much faster pixel manipulations by writing to the display memory directly, and indeed, many commercial packages do just that. The speed advantages of this approach, however, are obtained with a high risk of incompatibility: environments such as Windows and OS/2 are much more intolerant of direct display memory access.

In selecting the ROM BIOS tradeoff rather than directly writing to screen memory, I've deliberately opted for the more conservative approach on two grounds: first, the ROM BIOS calls are the approved method for doing graphics and future machines will probably support them within the definition of "well-behaved"; and second, faster processors will cancel out their slower speed, resulting in a zero loss/gain from the user's perspective. While this is an arguable position, and no doubt some readers will dispute it, at least you know my underlying assumptions. Now let's see how to access those ROM BIOS functions.

ROM BIOS Calls from Turbo C

Like DOS itself, the ROM BIOS routines under interrupt *10h* expect a function code in register *AH*, with other parameters passed as required by specific functions in the rest of the registers. Any number of Microsoft, IBM, and commercial publications document the ROM BIOS calls; for this project I referred to Ray Duncan's indispensable *Advanced MS-DOS* (pages 399–420).

ROM BIOS calls are made using the Turbo C *int86( )* function. Registers are set up in a structured variable bound to the *REGS* union as defined in Turbo C's DOS.H header file.

The *REGS* union includes all the general registers of the 80x86 line of processors and furnishes a notation convention for byte (8-bit) and word (16-bit pair) registers. For example, if you declare the structured variable as *union REGS r;* you can load the value *0Ch* into byte register *AH* with *r.h.ah=0x0C;* and the same value into word register *BX* with *r.x.bx=0x0C;*. The structure notation *.h.* indicates a byte register, and notation *.x.* indicates a word register. The *REGS* union encompasses all byte registers (*AH–DL*) and all word registers (*AX–DX*) as well as *SI*, *DI*, and the flags. It does not include the segment registers (*CS*, *DS*, *ES*, and *SS*), which are irrelevant to ROM BIOS calls.

To call a ROM BIOS video service

Kent Porter, 1909-4 Montecito Rd., Mountain View, CA 94043. Kent has written 17 books about programming and hundreds of magazine articles on computer hardware and software. He is a technical editor for DDJ.

routine, you place the function code in *r.h.ah* and the required parameters in other registers and execute the Turbo C *int86( )* function *int86 (0x10, &r, &r);*.

Function *int86( )* performs the following:

- saves the caller's registers
- loads the CPU registers from the union whose address is passed in the second argument
- executes the interrupt given in the first argument
- on return from the ROM BIOS, places register contents in the union passed as the second address argument (the same as the first address argument in this example)
- restores the caller's registers
- resumes execution in the calling program

On resumption you can pluck any BIOS-returned value from the variable bound to the *REGS* union—*r* in the examples shown here—by using the appropriate register-type notation.

For example, suppose you want to determine the current cursor position in video page 0. The setup is:

```
r.h.ah = 0x03; /* read position */
r.h.bh = 0; /* in page 0 */
int86 (0x10, &r, &r); /* call BIOS */
```

Afterward you can fetch the row with *cursRow=r.h.dh;* and the column with *cursCol=r.h.dl;*. This is the same as, but easier than, performing an equivalent call in assembly language. An added benefit is that the structured variable bound to the *REGS* union retains the returned register values for as long as it exists or until the next call to the ROM BIOS routines. Thus, you can fetch the returned register values at your convenience. You can translate these ROM BIOS calls into C functions.

Syntactic Sugar

Many of Turbo C's 350 functions and macros are simply DOS and ROM BIOS calls sugar-coated to look like C. In the same spirit, you can pack a little sugar around the ROM BIOS video service routines and call them "a basic library of Turbo C graphics functions."

Listing One, page 82, shows an *#include* file, VIDEO.I, which contains the fundamental 16 ROM BIOS calls plus a couple of extended functions that synthesize several calls. Any program that has to perform graphics operations can obtain the full set of functions with the simple statement *#include <video.i>* calling whichever specific ones it needs. Because the source code is included in the compilation, the functions will automatically adapt to the memory model currently in use. Also note that every function is defined before being referenced by other functions, thus eliminating any need for a header file containing prototypes (if you program using ANSI C conventions).

The down side of using source-level *#include* files such as VIDEO.I is that they waste memory. Turbo C doesn't optimize itself by pulling out unreferenced code. As a result, all the code for all the functions appears in every .EXE program that *#includes* VIDEO.I, even if the program calls only a single function. My solution to this is to make the functions into a linkable library (.LIB file).

Creating the Libraries

Building and linking with user-created libraries in Turbo C can be a bit of a chore. In the small model, the inclusion of VIDEO.I adds 1,376 bytes to the .EXE file; in the large model, 1,520 bytes. The overhead consists of total bytes minus the sizes of the functions you actually call. If that amount of code space is important to you, it might be worth turning VIDEO.I into one or more libraries so that only those routines that are actually used get linked into the .EXE file.

The first step is to write a file VIDEO.H that defines all the graphics function prototypes. Listing Two, page 84, shows this header file, which you should *#include* in any programs that link with the video library. You'll also use VIDEO.H in creating the library's individual members. Each function file should *#include* VIDEO.H at the top. Purists might also want to add comments explaining what the function does and to what library it belongs.

When all the functions have been broken out into separate files, compile them one by one with Turbo C to create a matching set of .OBJ files. Now you're ready to make the library.

The .OBJ files produced by Turbo C are in Microsoft-compatible format, so you can use the DOS LIB utility to combine them into a linkable library. To produce a library called VIDEOS.LIB containing the functions video *mode( )* and *activepage( )*, for example, type:

```
LIB VIDEOS + VIDEOMODE + ACTIVE-
                          PAGE;
```

Later you can add *setmode( )* and *setcursor( )* with the similar:

```
LIBVIDEOS + SETMODE +
                    SETCURSOR;
```

You can, of course, add more than two modules at a time to the library simply by specifying the library name first, followed by an entire command line of module names separated by plus signs.

The *S* on the end of the library name is a convention borrowed from Turbo C to indicate the memory model; here, it is assumed that you compiled the separate units in the small model.

To link with this library, first make sure it's in the directory where Turbo C looks for source files (not in the \TURBOC\LIB directory with the vendor's libraries). Put the entry VIDEOS.LIB in your project file (.PRJ). Turbo C then knows that you want to link to a user-created library and where to find it.

Repeat the compilation and LIB procedures for each memory model you use, varying the library file name accordingly (VIDEOT.LIB, VIDEOC.LIB, and so on). You'll have to modify the library name in your .PRJ file if you decide to change memory models during the development of a program.

Because it takes an hour or two to get through this whole business of making libraries, weigh the price in time investment vs. saving a thousand bytes or so in the .EXE file. It's your choice.

Adapting to the Adaptor

The box accompanying this article

GRAPHICS TOOLBOX

(continued from page 31)

(page 34) describes how to determine which video adaptor is present in the machine. Refer to it for the "theory"; this article concentrates on the practical aspects of applying the adaptor identification to APA graphics, specifically on the CGA and EGA. The principles discussed here are applicable to other adaptors, such as the Hercules, as well.

In text modes there's little adapting to do for a specific display type. The screen is always 25 rows high and either 40 or 80 columns wide. The 40-column mode only applies when writing text on a 320×200-pixel graphics screen; otherwise you're always in 80-column mode. The monochrome display adaptor (MDA) can't produce colors, but it can do normal, intense, underlined, and blinking (attributes remarkably similar to colors). Thus, although the demo program later in this article does a few text tricks using the video library, I'm not going to discuss them here. Because text graphics has its own set of problems and solutions, I'll save that discussion for Part 2.

When your software knows the video adaptor it's working with, it can alter its behavior to suit. After identifying the video board, the software can then adjust its coordinate system or color selections to fit the APA display. Here, in the interest of limited space, I'll show you how to make a program dynamically modify its coordinate system.

Suppose, for example, you want to draw a border around the screen, starting 15 scan lines (y points) below the top and ending 15 above the bottom. The width in either case is 640 pixels and the top of the rectangle is at y=15. However, the CGA has a mere 200 vertical scan lines, whereas the EGA has 350. Therefore, you can set up an assignment condition such as:

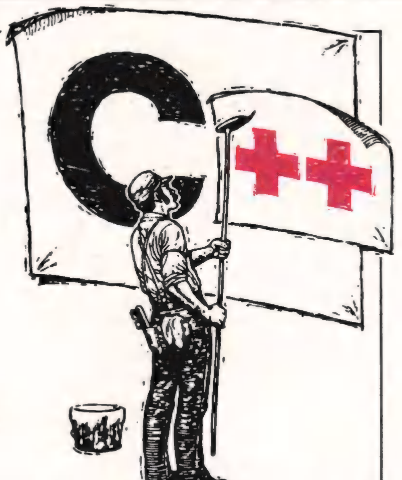
```
if (adaptor = = cga)
   bottom = 199–15;
else
   bottom = 349–15;
```

The trick here is to locate the bottom of the box, which is determined by the adaptor type. The vertical line-drawing routine can then repeatedly call the *plot()* function from the video library, with the number of calls—pixels plotted—being determined at run time by the type of adaptor available. Similarly, the horizontal line-drawing routine can draw the bottom at the appropriate elevation (y=184 or y=334, for CGA and EGA, respectively).

This is a somewhat simplistic scenario implemented in the demo program; a more complex application might consider CGA the default and supply a multiplier of 1.0 in calculating y coordinates. If an EGA is present, however, the program can substitute 1.75 for the y multiplier (because 350/200=1.75). Furthermore, if four-color graphics are required, it can use 1.0 as the multiplier for the default (mode *04h*=CGA 320×200 four color) and substitute 2.0 if an EGA is present (mode *10h*=EGA 640×350 four color). Thus the program dynamically redefines its coordinate workspace in both dimensions based on the available video adaptor.

A program that does extensive APA graphics will rely on the video library's *plot()* function, which plots

Now for VAX C & Sun C



Add C++ to your favorite C Compiler

- **Object-oriented C**
- **Strong type-checking**
- **Works with your present C Compiler**

DESIGNER C++

BENEFITS:

► Works with the C Compiler you now use

► You can incrementally add C++ features to C (switch-selectable)

► Makes C more suitable for
— very large programs
— more sophisticated applications

► More reusable code

► Resilient and bug-free code

FEATURES:

- Fully compatible with AT&T C++ standard
- Optional strong type checking
- Data abstraction
- Overloading of function names and operators
- Dynamic typing (virtual functions)
- User-defined implicit type conversion
- Works with Sun's *dbxtool*

The only commercially-available C++ customized to operate on PC's, micros, minis, and mainframes with popular C compilers, including:

VAX C	SUN C
ULTRIX C	MICROSOFT*
APOLLO	LATTICE*
XENIX	GREEN HILLS
HP-9000	UNISOFT

*Lattice and Microsoft versions of Designer C++ are known as Advantage C++.

We Specialize in: Cross/Native Compilers: C, Pascal, FORTRAN, Ada, LISP — Assemblers/Linkers — Symbolic Debuggers — Simulators — Interpreters — Profilers — QA Tools — Design Tools — Comm. Tools, — OS Kernels — Editors — VAX & PC Attached Processors and more
We Support: 680xx, 80x86, 320xx, 68xx, 80xx; Clipper, and dozens more

A DIVISION OF XEL **Oasys**

230 Second Avenue, Waltham, MA 02154 (617)890-7889

Designer C++ is a joint trademark of XEL, Inc. and Glockenspiel, Ltd of Dublin. Ada is a trademark of the U.S. Government (AJPO). Advantage C++ is a trademark of Lifeboat Associates, Inc. Other trademarks are acknowledged to DEC, Lattice, Microsoft & Sun Microsystems, Inc.

CIRCLE 254 ON READER SERVICE CARD

Once again,
Compaq
raises the standard
of performance
for personal computers.

This time
by a factor of two...

Introducing the two on earth

The new COMPAQ DESKPRO 386/20™

Last year, we introduced the COMPAQ DESKPRO 386,™ the most advanced personal computer in the world. Now the world has two new benchmarks from the leader in high-performance personal computing. The new 20-MHz COMPAQ DESKPRO 386/20 and the 20-lb., 20-MHz COMPAQ PORTABLE 386 deliver system performance that can rival minicomputers.' Plus they introduce advanced capabilities, without obsoleting your investment in software, hardware and training.

Our new computers employ an industry-standard 20-MHz 80386 microprocessor and sophisticated 32-bit architecture. But to make these two of the world's fastest PC's, we did more than just increase the clock speed.

For instance, both are built around a concurrent bus architecture. Two buses—one for memory and one for peripherals—eliminate information bottlenecks, allowing each component

It simply works better.

most powerful PC's
and off.

and the new 20-MHz COMPAQ PORTABLE 386™

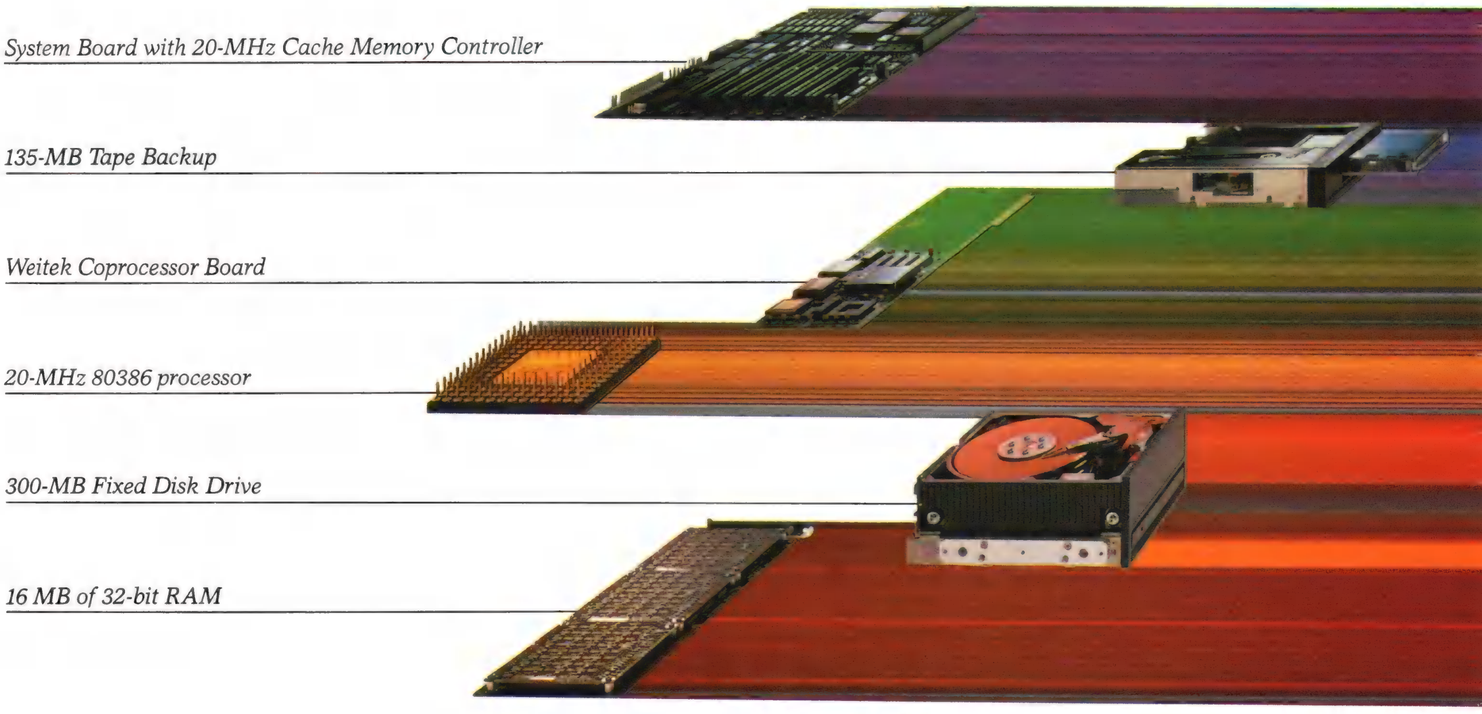
to run at its maximum speed. Together, they insure the highest system performance without sacrificing compatibility with industry-standard peripherals.

Both computers offer disk caching. Both offer the most memory and storage within their classes. Both let you run software being written to take advantage of 386 technology. And both run new MS-DOS®/BASIC Version 3.3 as published by Compaq. With it, our new portable and our new desktop can break the 32-megabyte limit on file sizes that handcuffs other PC's, allowing you to build files up to the size of your entire fixed disk drive.

And from now until December 31, 1987, both computers come with a free package of new Microsoft® Windows/386 Presentation Manager. It provides multitasking and switching capabilities with today's DOS applications to make you more productive. But that's just the beginning. To find out more, read on.

COMPAQ®

The question wasn't but how to get the



System Board with 20-MHz Cache Memory Controller
135-MB Tape Backup
Weitek Coprocessor Board
20-MHz 80386 processor
300-MB Fixed Disk Drive
16 MB of 32-bit RAM

The most powerful personal computer in the world

The COMPAQ DESKPRO 386/20 is an impressive 50% faster than 16-MHz 386-based personal computers. Even more impressive is the fact that it's up to 25% faster than other 20-MHz 386's. That's because the processor is just one small part of how the COMPAQ DESKPRO 386/20 outperforms every other PC in the world today and even many minicomputers.

The big reason is the new COMPAQ Flexible Advanced Systems Architecture, which optimizes overall system throughput while maintaining full compatibility with industry-standard peripherals. It does this by combining an advanced memory caching scheme with memory and peripheral buses that operate concurrently.

Complementing the speed of the microprocessor is the new advanced 20-MHz Intel® 82385 Cache Memory Controller. Like an efficient secretary that keeps frequently used information close at hand, it allows the microprocessor to operate at 0-wait states 95% of the time.

While one bus handles these high-speed operations, another *simultaneously* handles periph-

It simply works better.

how to get to 20 MHz, most out of 20 MHz.

erals operating at the industry-standard 8 MHz.

This flexible approach allows you to dramatically increase system throughput while preserving your investment in monitors, disk drives, and expansion boards. It can also accommodate today's and tomorrow's most advanced peripherals without constraining their performance.

Take options like our new Weitek™ Coprocessor Board. Never before offered in a PC, it can increase the speed of calculation-intensive, engineering and scientific applications by a factor of six, giving the COMPAQ DESKPRO 386/20 the performance of a dedicated engineering workstation at a fraction of the cost.

Compaq also provides 130- and 300-Megabyte Fixed Disk Drives with some of the industry's fastest access times. And when used with disk caching software, they represent the highest-performance storage subsystems available.

As for memory, Compaq offers 32-bit high-speed RAM. One full megabyte comes standard and is expandable to 16 megabytes without using an expansion slot. Plus, we included the COMPAQ Expanded Memory Manager. It supports the LIM standard so your software can break the 640-Kbyte barrier even before OS/2™ is released.

As tasks become more complex and users demand more advanced capabilities, Compaq responds by raising the standard of performance in personal computing.

COMPAQ

DESKPRO 386/20™

Everyone expected Compaq
But no one

Pound for pound, it is the world's most powerful computer

Compaq has long been recognized as the world leader in both 80386 technology and portable computing. So it isn't surprising that we would combine the two.

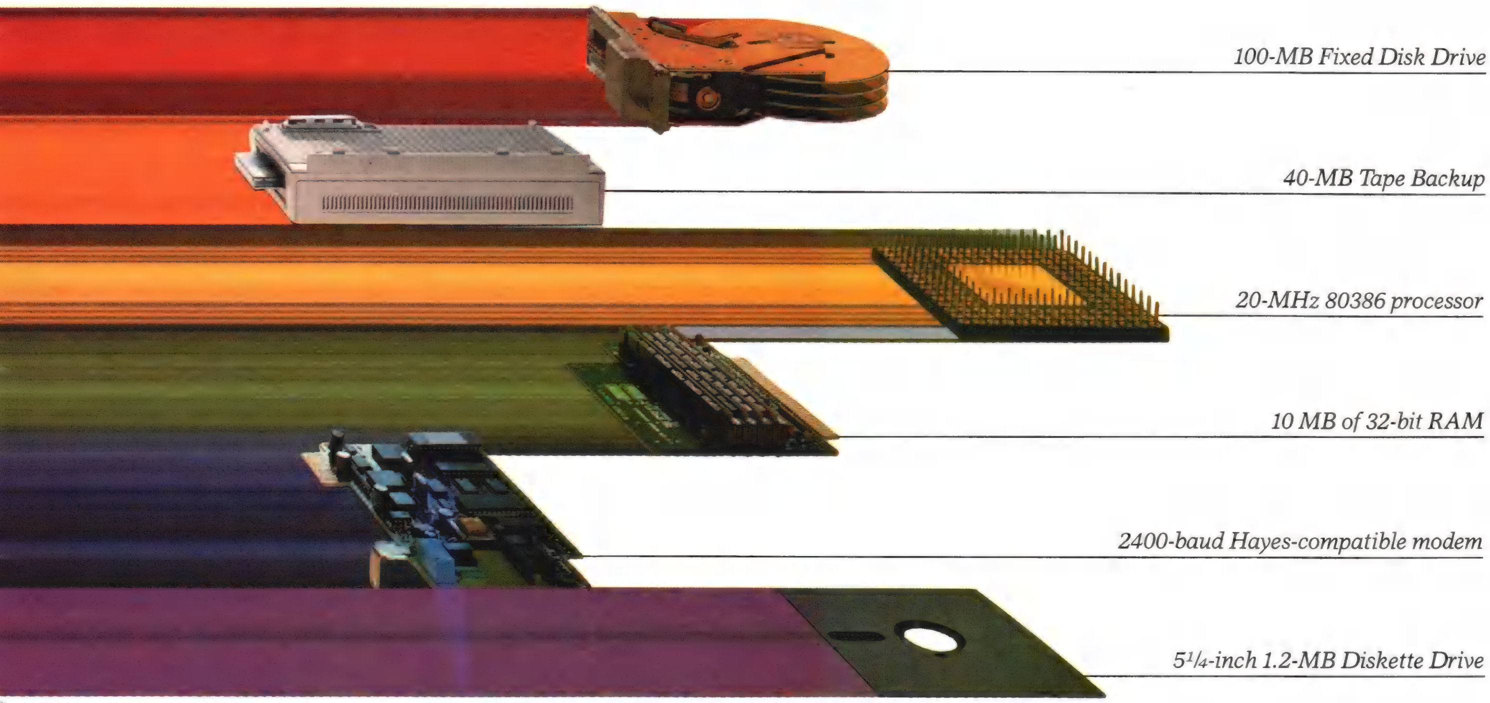
But no one expected the new COMPAQ PORTABLE 386 to run at 20 MHz. And no one even dreamed that it would offer 100 megabytes of storage, disk caching, and much, much more.

Our newest 20-lb. portable computer goes far beyond an 80386 microprocessor with a handle. It's not just the most advanced portable in the world. Pound for pound, it's the world's most powerful computer. Period.

Like the recent COMPAQ PORTABLE III,™ which changed the shape of full-function portable computing, the COMPAQ PORTABLE 386 makes no compromises. It offers more speed, memory, storage and features than any other portable PC. It runs your current software up to 25% faster than 16-MHz 386 PC's. Beyond that, its performance in calculation-intensive

It simply works better.

to introduce a 386 portable PC. expected all this.



applications is increased even more when you add an optional 20-MHz 80387 coprocessor.

Memory? Get one megabyte of 32-bit, high-speed RAM standard or go as high as 10 MB internally. And like all of the COMPAQ 386-based PC's, it features the COMPAQ Expanded Memory Manager.

With our high-performance 100-megabyte internal fixed disk drive, you can actually fit 500 lbs. of data-filled pages into a 20-lb. PC, unsurpassed storage for a portable. If that's too much for you, we also offer a 40-megabyte model.

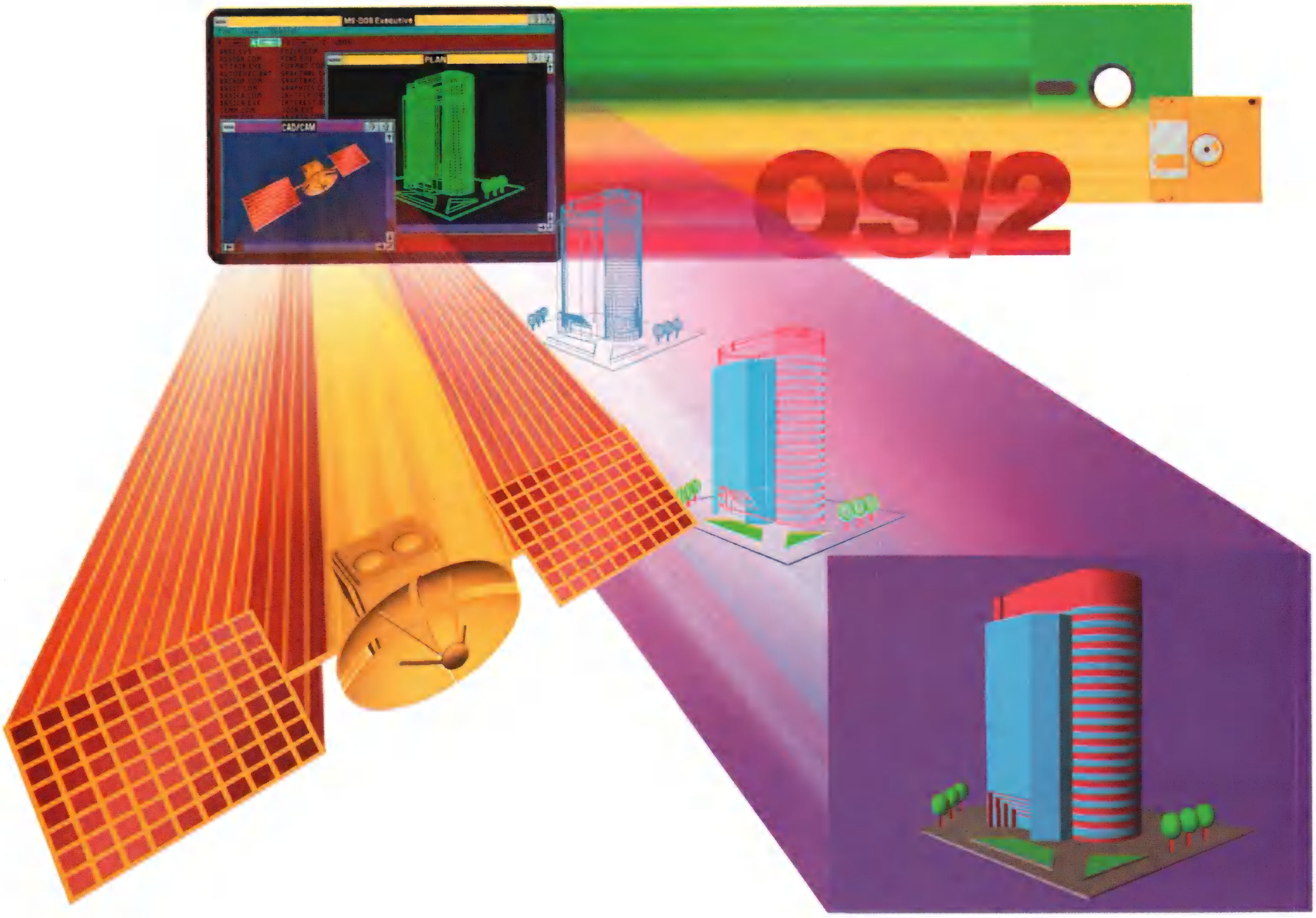
We've become famous for building desktop computer capabilities into our portables without leaving anything out. The COMPAQ PORTABLE 386 is more proof. It has a high-resolution, 640 × 400, 10-inch plasma display; a full-size, portable enhanced keyboard; two industry-standard expansion slots in a lightweight, optional plug-on unit; a choice between an optional 2400- or 1200-baud Hayes®-compatible modem; a full-size 5¼-inch 1.2-MB diskette drive; even an optional 40-MB tape backup.

These features, combined with the ultimate in portable performance, make the COMPAQ PORTABLE 386 the *biggest* PC this *small*.

COMPAQ

PORTABLE 386™

Compaq moves you ahead without leaving you behind.



Compaq offers the most complete line of high-performance 386 solutions. They all run industry-standard software and hardware, protecting the investments you've already made.

At the same time you won't be left behind when other technologies become important. Multitask with existing applications using Microsoft Windows/386 Presentation Manager. Add VGA graphics if you wish. Run OS/2 when it's available. And now 3½-inch drives are even an option for our desktops.

We optimize the most advanced technology while maintaining compatibility with the past, present and future. This makes COMPAQ PC's a wise decision for serious business users. Because at Compaq, we don't burn bridges, we build them.

See the COMPAQ DESKPRO 386/20 and COMPAQ PORTABLE 386 at an Authorized COMPAQ Computer Dealer. *And from now through December 31, 1987, get Microsoft Windows/386 Presentation Manager free when you buy a 386-based COMPAQ computer.* For more information, call 1-800-231-0900, Operator 40. In Canada, call 416-733-7876, Operator 40.

COMPAQ®

It simply works better.

GRAPHICS TOOLBOX

(continued from page 32)

individual pixels, but it could clearly benefit by some higher-level drawing routines.

Adding APA Graphics

Ultimately, all objects appearing on the display are composed of three kinds of lines: vertical, horizontal, and diagonal. Even a solid such as a block cursor or a complex filled polygon consists of some number of contiguous lines arranged so as to approximate a form that the eye recognizes. This suggests enhancing the video library with line-drawing routines that you can employ to create shapes.

For efficiency I developed orthogonal line routines and a separate, less efficient routine for diagonals.

There's less overhead in drawing orthogonal lines (vertical or horizontal) than in drawing diagonals. Why? Because diagonals proceed along a slope, or ratio between vertical and horizontal motion. Slopes are, almost without exception, fractional numbers that entail floating-point operations, which are inherently slower in computers. Slope is not an issue when drawing a line that is perfectly vertical or horizontal, and thus it can be done with more efficient integer arithmetic.

For efficiency, I developed orthogonal line routines and a separate, less efficient routine for diagonals. The latter is called if I know or suspect that the line is not orthogonal.

In the orthogonal routines, you pass the start and end points, the axis coordinate along which the line is to proceed, and the color of the pixel you want plotted. The routine then uses integer arithmetic to control a loop that plots the individual points. Listing Three, page 84, supplies two such routines, called *hdraw( )* and *vdraw( )*, for horizontal and vertical lines, respectively.

Because callers will probably use these routines frequently, passing variables as arguments, you should not place on them the burden of ordering the start and end points to suit the expectations of the routines. Consequently these routines sort the start and end into the low-to-high order they expect. The calls *hdraw (35, 231, 20, 1);* and *hdraw (231, 35, 20, 1);* both produce exactly the same result: namely, a horizontal line between x coordinates 35 and 231 inclusive along y coordinate 20 in color index 1.

In considering the diagonal routine *draw( )*, it's important to recognize several factors:

- The requested line might be orthogonal, resulting in a slope that is either 0 (horizontal) or infinity (vertical). To avoid division-by-zero failures, the routine must anticipate these problems in advance and take corrective action by preassigning the correct stepping value.
- The diagonal must move along the axis that results in the densest line, or in other words, along the axis that has the greatest distance to travel. If the motion is greater along the x axis than along the y, the more dense line results from plotting each y intersection per x increment. Thus the routine must determine the axis with the greatest movement and make that the controlling axis.
- The stepping value along the shorter-motion axis is a floating-point number that is the ratio of short-motion to long-motion. For example, if the x axis moves +120 points and the y axis –30, the x axis is controlling and the y moves –0.25 points per x (–30/120 = –0.25), which is its slope relative to the controlling axis.
- The stepping value is additive. That is, for each controlling increment, the stepping value is added to the current controlled value. This motion accumulates and is truncated or rounded for each controlling step, according to the way the routine is written (Listing Three truncates it).

Sometime next year, C will be 20 years old. If you're starting a project that will last into the next decade, you should at least pick a language designed in this one. We supply libraries for the safest choice:

MODULA-2

Our Products Include:

★ *Repertoire ®:* 5 high-level subsystems and hundreds of low-level routines for M2. Release 1.5 includes fast screen design/display system for virtual windows that scroll horizontally and vertically, plus multi-line input fields, help that's context sensitive to field level, menus, forms, etc. (not a code generator; keeps frames on disk until display). Also includes text editor and fast DBMS with variable-length keyed records, garbage collection, damaged file recovery, named and nested fields. Stores anything: bitmaps, text, linked lists, binary records, etc. Includes full source (over 600K) and 300 page manual. **$89**

★ *EmsStorage™* advanced, high-level memory manager that detects and uses LIM Expanded Memory if present, or DOS memory if not; lets users conserve scarce DOS memory for other programs, and lets larger programs run under Logitech's debuggers. Provides automatic garbage collection and MS- Windows-like interface (lock/unlock functions) for porting programs to MS Windows. **$49**

★ *Graphix:* the Modula-2 interface to MetaWINDOW, the fast professional graphics system PCTJ named 7/85 Product of the Month. Supports multiple fonts, over 30 display adapters, and hundreds of modes. Includes full MetaWINDOW package. **$149**

★ *ModBase:* a B+Tree DBMS that uses a file format compatible with Ashton-Tate's dBase III. Provides indexing and file manipulation routines for use independent of dBase; billions of records per file. Includes full source code. **$89**

★ *Macro2:* a macro preprocessor for Modula-2; provides inline expansion of functions, include files, conditional compilation, etc. Includes full source code: . **$89**

★ *Full source code for **Make** and **Xref** utilities; customize them to work exactly as you like. Each* **$89**

*All available exclusively from PMI; dealer inquiries welcome. Versions for Logitech, Stony Brook, FTL and ITC; all libraries available as DOS LINK-compatible object modules, useable by other MS-Standard languages. **Full documentation for these and many other products available on free demo disks.***

PMI

VISA/MC
AMEX/COD/PO

(503) 777-8844
4536 SE 50th — **BIX: pmi**
Portland, OR 97206 — **CIS: 74706,262**

CIRCLE 239 ON READER SERVICE CARD

Identifying the Video Adaptors of IBM PCs

ROM BIOS interrupt *10h* on IBM PCs and compatibles furnishes video services to accommodate a variety of hardware configurations. Although many of the calls—notably those dealing with text—are device independent, others are device specific. If your graphics software is to function effectively in this kind of variable environment, it must adapt its behavior to suit the hardware. And to do that, it must first find out what the video hardware is.

This isn't as easy as it might sound. Programs that want to determine the video capabilities at their disposal usually have to look in several places, piecing together many clues.

Value	Adaptor
0	undefined[1]
1	40 by 25 B&W text, color graphics[2]
2	80 by 25 text, CGA[3]
3	80 by 25 text, monochrome

1. used by Compaq, EGA, and CGA
2. also used for non-IBM video devices such as Hercules
3. unreliable

Table 1 *BIOS video equipment flag*

Mode	Meaning
00h	40 by 25 B&W text, color graphics (obsolete)
01h	40 by 25 color text
02h	80 by 25 B&W text
03h	80 by 25 color text
04h	320 × 200 4-color graphics
05h	320 × 200 4-color graphics (color burst off)
06h	640 × 200 2-color graphics
07h	monochrome adaptor or EGA text mode
08h	160 × 200 16-color graphics (PCjr)
09h	320 × 200 16-color graphics (PCjr)
0Ah	640 × 200 4-color graphics (PCjr)
0Bh	not used
0Ch	not used
0Dh	320 × 200 16-color graphics (EGA)
0Eh	640 × 200 16-color graphics (EGA)
0Fh	640 × 350 2-color graphics (EGA)
10h	640 × 350 4-color or 16-color graphics (EGA) (depends on available display memory)

Table 2 *IBM PC line video modes*

The Video Equipment Flag

First I'll discuss sources of information about the video environment, then I'll show how to use them to identify the video hardware. At memory address *40h:07h*, the ROM BIOS keeps a byte called the equipment flag, which describes the machine's hardware configuration. Bits 4 and 5 give some information about the attached video device. By masking out the other bits and shifting right four places, you can isolate a device number from 0 to 3. In C, for example, you might write this operation as:

```
adaptor = (peek (0x40, 0x07) & 0x18)
                               >> 4;
```

A similar effect can be obtained in BASIC with the statements:

```
DEF SEG &H40
ADAPTOR = (PEEK (7) \\ 8) AND
                          &H03
```

Table 1, left, shows the full range of resulting values. As this table indicates, the BIOS video equipment flag byte raises more questions than it answers. The only thing that's sure is that when its value is 3, you have a monochrome (that is, text-only) adaptor. Even so, you might want to check further; if the video mode discussed next is 7, it's definite that the video device is incapable of APA graphics and color text.

The Video Mode

The ROM BIOS furnishes two methods for obtaining the video mode—that is, the current mode of operation for the adaptor. One is to call interrupt *10h* with function code *0Fh* in the *AX* register. On return, *AL* contains the mode value. An alternative is to bypass the ROM BIOS and do yourself what function *0Fh* does indirectly: read the video mode byte from location *40h:49h*.

Whichever, the resulting is a byte whose possible settings are shown in Table 2, left. These values can also be used to set the video mode, assuming the attached device is capable of supporting them. These values suggest the capabilities of the monitor by inference only and can be misleading. For example, although mode *07h* indicates an active monochrome display adaptor (MDA), it is also valid for the EGA in plain text mode.

Note that modes *0Bh* and *0Ch* are undefined; presumably these are reserved for future developments or for graphics adaptors that were never introduced.

The Enhanced Graphics Adaptor

If the current mode is anything less than *08h*, you have to investigate further to find out if an EGA card is present in the system. The ROM BIOS furnishes no call for this, so you have to read memory location *40h:87h* to find out. This address contains the EGA information byte.

Although each bit or bit pair in the EGA information byte has some unique significance, in general it's sufficient to know that the byte is 0 when an EGA is not present and nonzero when one is. Therefore, in C, you can define a Boolean function *isEGA( )* as follows:

```
char isEGA (void)
{
  return (peekb (0x40, 0x87));
}
```

which returns *FALSE* when there is no EGA attached and *TRUE* (some nonzero) when one is. Similarly, you can test for an EGA monitor in BASIC with statements such as:

```
DEF SEG &H40
IF PEEK (&H87) <> 0 THEN (EGA
              present) ELSE (not)
```

A brief discussion of the EGA is perhaps a worthwhile digression here. The EGA furnishes several extended all-points addressable (APA) graphics capabilities. That is, you can have more pixels of higher density with more colors than in the earlier graphics adaptors. How the EGA works is beyond the scope of this article. What's important is that the EGA is downward compatible; that is, it supports all the modes available in the earlier MDA and CGA, plus some of its own. The EGA offers no text capabilities beyond the CGA, except that the characters are more dense and thus more readable; this is a given, beyond programmer control. Otherwise the same combinations of 16 foreground by 16 background colors are available.

The advantages of the EGA over other adaptors thus involve only APA graphics. In text modes, the only discriminator is whether the display is pure monochrome (MDA) or not. If it's pure MDA, you can't do colors; otherwise you can.

Other Common Adaptors

The popular Hercules card offers APA graphics enhancements but nothing extraordinary by way of text. The Hercules provides for gray shading of text as a substitute for text colors and APA graphics on a 720 × 348-pixel monochrome display.

The Compaq adaptor is second only to the EGA in versatility. In text mode, it acts like the EGA's text mode, producing high-resolution text in any combination of 16 foreground and 16 background colors. You can also switch it into any of the CGA graphics modes (*04h–06h*) and it behaves just like the CGA adaptor.

Identifying a "Herc" or a Compaq, like the others, is a matter of recognizing a pattern of indicators.

The Software Sherlock Holmes

Now let's see how these clues fit together. Table 3, below, shows the "signatures" of several popular video adaptors in their default (power-up) conditions. Notice that each adaptor has a unique pattern of values. Given this, you can quickly sift through three items of information and identify the adaptor. Expressed in pseudocode, the algorithm is:

```
if EGA byte <> 0 then
  adaptor = EGA;
else
  case BIOS equipment flag of
    0: if video mode = 2 then
         adaptor = Compaq;
       else
         adaptor = CGA;
       end if;
       break;
    1: adaptor = Hercules; (see note)
       break;
    3: adaptor = MDA;
  end case;
end if;
```

Note: Some other monitors use the same signature as the Hercules does. An example is the MDS Genius full-page display popular with desktop publishing systems. In this case, you have to look at the display buffer size, which you can fetch as an integer from *40h:4Ch*. The Hercules display buffer is 16,384 bytes; that for the Genius is 8,192 bytes.

Obviously, then, Table 3 and the algorithm are not comprehensive. The great majority of video adaptors for IBM PCs and compatibles emulates one of these de facto standards, however, and thus the method presented here is reliable in almost all cases.

	EGA	CGA	MDA	Compaq	Hercules
BIOS equip.	0	0	3	0	1
Video mode	3	3	7	2	7
EGA byte	nonzero	0	0	0	0

Table 3 *Signatures of popular video adaptors*

Introducing Network Version!

SEIDL VERSION MANAGER

Now SVM supports local area networks and tracks source revisions made by multiple users in both single-site and multi-site configurations.

Plus...

- Archive Database Tracks Source (and Binary) File Revisions
- Audit Trail Reporting Provides Info on Project's Development
- Revision Branches Allow Multiple Courses of Development
- Revision Merging and Deleting Provide Flexibility in Archive Maintenance
- User IDs, Privilege Settings & Passwords Help Resolve Access Conflicts and Maintain Project Integrity
- Optional Text Compression Reduces Storage Requirements
- Menu Driven Shell Makes SVM Easy to Use
- Single-Site: \$299.95*
- 5-site LAN: \$1000 (extendible)

Now Builds Dependencies!

SEIDL MAKE UTILITY

New program, called SMKgen, automatically constructs a dependency file by analyzing the files in a project.

Plus...

- Structured Language Used to Define Dependencies
- Rich Command Set with Over 20 Different Statements
- Ability to Handle Nested Include Files and Library Dependencies
- Performance & Functionality not Found in UNIX Make or Clones
- SMK Only: \$99.95*
- SMKgen: Add \$50.00

CALL TODAY

1-313-662-8086

Visa/MC/COD Accepted
Dealer Inquiries Invited
*Plus postage and Handling

SEIDL COMPUTER ENGINEERING

3106 Hilltop Dr., Ann Arbor, MI 48103

CIRCLE 114 ON READER SERVICE CARD

GET THE BASIC FACTS

Introducing ApBasic™ the only program available today that offers a <u>real</u> debugger feature, with single-step, watch variables and breakpoints.

Plus, on-line help for language, one megabyte of code space and string space, a true native code compiler (no psuedo-code) and structured code. And, because it's ApBasic, there's more!

Editor Features • *Fast full screen editor with undelete lines, block copy and move, search and replace, one keystroke to compile and run* • **Language Features** • *8087/80287 support • Alphanumeric line labels, no more line numbers • Multi-line if statements • Supports DOS 3.xx networks • Named constants • Text windows • Modular sub-programs and multi-line functions with local, static and global variables up to 64K of code each* • **Debugger Features** • *Up to twenty breakpoints • Up to ten watch variables • View output screen while in debugger • single-step or go to line with cursor* • **System Features** • *Source code toolboxes available soon • Creates stand-alone .EXE files or smaller chain files • Compact compiler, runs easily on a single diskette computer.*

INTRODUCTORY OFFER
$69.95
OFFER EXPIRES DEC. 31, 1987

When you get the facts, there <u>is</u> a basic difference.

ApBasic
BY COMPTECH

1 • 904 • 497 • 4810

Comptech Software and Consulting, Inc.
P.O. Box 280, Ft. White, FL 30238
TELEX 910 • 250 • 4832 COMTECH UQ

CIRCLE 94 ON READER SERVICE CARD

GRAPHICS TOOLBOX
(continued from page 33)

These considerations enable the routine to plot a diagonal or orthogonal line in any direction. The cost of this flexibility is slower performance.

As you use the video library, you'll undoubtedly add your own graphics routines for such things as polylines, filled shapes, circles, arcs, and so on. The DRAW.I file given here is merely a suggestion of the kinds of things you can do to make graphics easier in Turbo C.

You can also make life a little easier, and your source code more readable, by including the file COLORS.H (see Listing Four, page 85) in programs that require colors. It's always easier to understand identifiers than "magic numbers" requiring you to memorize, for example, that *07h* is light gray.

A Graphics Sampler

Now let's put this discussion to work in a demonstration program. Listing Five, page 85, contains the demonstration program VID.C, which calls a number of the routines in the video library. It produces some simple but illustrative effects in both text and graphics modes.

The program first identifies the adaptor on the host machine so that it can subsequently tailor its behavior to suit (or bypass operations that the adaptor can't handle, such as APA graphics on an MDA). This program is written to run on the MDA, CGA, EGA, and Compaq; it doesn't recognize the Hercules card.

Next it produces a text graphics display that showcases cursor positioning with *gotoxy( )* and (if the monitor is capable of color) colored text. Except for the string-writing functions that produce a label at the top and a prompt at the bottom of each demonstration display, this is the only use of text graphics in the program.

In the third display, the program draws a border around the screen and a large corner-to-corner X inside it. It calls the functions in DRAW.I. This routine shows how to make a run-time adjustment to the coordinate system according to whether the adaptor is a CGA (or Compaq) or an EGA. If you're running with an MDA, the program bypasses this display and the next because the hardware can't handle it.

The fourth panel is in CGA 320 × 200-pixel four-color graphics. It draws a three-color hourglass illustrating two things: first, that solids on a display are indeed composed of numerous horizontal lines; and second, that the EGA (if that's what you're using) can emulate a CGA monitor.

The final display merely announces that the demo is finished.

If you haven't converted the video functions into a linkable library but instead are using Listing One directly, change the indicated directive to *#include <video.i>* near the top of the file before compiling. On the other hand, if you've gone to the trouble of creating VIDEO.LIB, set up a project file VID.PRJ containing the following entries:

```
vid
video.lib
```

Make sure VIDEO.LIB is in the same directory as your source program VID.C, then start the compile, link, and go session with Alt-R.

Availability

All the source code for articles in this issue is available on a single disk. To order, send $14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call (415) 366-3600, ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

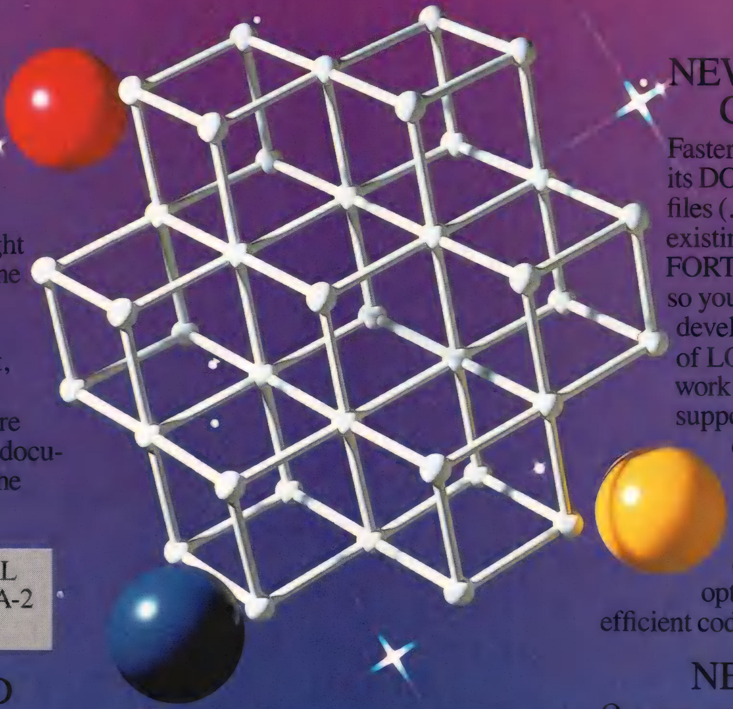
DDJ

(Listings begin on page 82.)

Vote for your favorite feature/article.
Circle Reader Service **No. 2.**

SOFTWARE ENGINEERING COMES OF AGE.

ANNOUNCING LOGITECH MODULA-2 VERSION 3.0



Modula-2 is the language of choice for modern software engineering, and LOGITECH Modula-2 is the most powerful implementation available for the PC. The right language and the right tools have come together in one superior product. Whether you're working on a small program or a complex project, with LOGITECH Modula-2 Version 3.0 you can write more reliable, maintainable, better documented code in a fraction of the time at a fraction of the cost.

FREE TURBO PASCAL TO LOGITECH MODULA-2 TRANSLATOR

NEW, IMPROVED DEBUGGERS

Time gained with a fast compiler can be lost at debug time without the right debugging tools. With the powerful Logitech Modula-2 Debuggers you can debug your code *fast,* and dramatically improve your overall project throughput.

The Post Mortem Debugger analyzes the status of a program after it has terminated while the dynamic, Run Time Debugger monitors the execution of a program with user-defined breakpoints. With their new, mouse based, multiple-window user interface these powerful debugging tools are a pleasure to use.

NEW, INTELLIGENT LINKER

Links only those routines from a particular module that you need, so you eliminate unreferenced routines and produce smaller, more compact executable files.

NEW, IMPROVED COMPILER

Faster and more flexible. Now its DOS linker compatible object files (.OBJ) can be linked with existing libraries in C, PASCAL, FORTRAN and ASSEMBLER — so you can build on previous development and put the power of LOGITECH Modula-2 to work for you right now. Fully supports Wirth's latest language definition, including LONGINT and LONGSET, which provides large set support including SET of CHAR. Provides optimization for tighter, more efficient code generation.

NEW EDITOR

Our new, mouse based editor is fully integrated, easy to learn, fast and easy to use, and very customizable. Its multiple, overlapping windows and color support make it easy to manage parts of one file or several files on the screen at one time. You'll love using it — with or without a mouse.

Call for information about our VAX/VMS version, Site License, University Discounts, Dealer & Distributor pricing.

To place an order call toll-free:
800-231-7717
In California:
800-552-8885

☐ LOGITECH Modula-2 V. 3.0 Compiler Pack **$99**
Compiler in overlay and fully linked form, Linkable Library, Post Mortem Debugger, Point Editor

☐ LOGITECH Modula-2 V. 3.0 Toolkit **$169**
Library sources, Linker, Run Time Debugger, MAKE, Decoder, Version, XRef, Formatter

☐ LOGITECH Modula-2 V. 3.0 Development System **$249**
Compiler Pack plus Toolkit

☐ Turbo Pascal to Modula-2 Translator **FREE**
With Compiler Pack or Development System

☐ Window Package **$49**
Build true windowing into your Modula-2 code.

☐ Upgrade Package
Call LOGITECH for information or to receive an order form.

Add $6.50 for shipping and handling. California residents add applicable sales tax. Prices valid in U.S. only. Total Enclosed $________

☐ VISA ☐ MasterCard ☐ Check Enclosed

Card Number ________ Expiration Date ________
Signature ________
Name ________
Address ________
City ________ State ________
Zip ________ Phone ________

LOGITECH

LOGITECH, Inc.
6505 Kaiser Drive, Fremont, CA 94555
Tel: 415-795-8500

In Europe: LOGITECH, Switzerland
Tel: 41-21-87-9656 Telex 458 217 Tech Ch

In the United Kingdom: LOGITECH, U.K.
Tel: 44908-368071 Fax: 44908-71751

Turbo Pascal is a registered trademark of Borland International. VAX and VMS are registered trademarks of Digital Equipment Corp.

CIRCLE 257 ON READER SERVICE CARD

A Graphics Toolkit for Turbo Pascal

by Hubert D. Callihan

Saving and restoring graphics screen regions

Although I've been using Turbo Pascal for custom graphics work for some time, I haven't come across much information on how to uncover the sometimes subtle graphics capability in Version 3.0. This article shows how I used several nonstandard Turbo Pascal procedures, including *GETPIC, PUTPIC, GETMEM, FREEMEM, BLOCKWRITE*, and *BLOCKREAD*, to create a set of tools for handling rectangular regions. You can use these tools, for which I have included all the source code, to develop:

- graphics pop-up overlay screens in either medium-resolution (320 × 200 pixels) or high-resolution (640 × 200 pixels) graphics modes
- multiple concurrent graphics regions that require rapid display
- rapid block-move animation

This set of tools consists of three procedures—*SaveRegion*, *RestoreRegion*, and *FreeBuffer*—along with one function—*CRTmode*—which collectively let you store to buffer and restore complete graphics screen regions or any upright rectangular subregions of the screen. I have also included two supporting procedures—*SaveBlockToDisk* and *GetBlockFromDisk*—which are useful for moving buffered screen regions to and from disk efficiently. (You need a CGA to make use of these routines.)

Dr. Callihan is chair of the Computer Science Department at the University of Pittsburgh at Johnstown where he has taught for the past 19 years. For the last 7 years, he has been actively involved in developing graphics tools.

Despite what you might think about the limitations of Turbo Pascal, the number of graphics regions that you can save to buffer simultaneously is limited only by heap space (that is, the dynamically accessible memory not already consumed by DOS, resident programs, the Turbo Pascal environment, the static program code, and static data). By operating in dynamic memory space, these tools exploit all the memory available on the PC that is managed by DOS. Turbo Pascal programmers who think they must stay within the 64K static limit will appreciate this feature; frankly, most of the criticism that has been leveled at Turbo Pascal regarding the 64K static limit is unfounded because, for a little inconvenience in notation and the need to allocate and deallocate memory, you can always get around the static limitation by using pointers.

I have successfully used these tools to develop menu overlays, images of fractals, Mandelbrot and Julia sets, splines, and custom animation code. Animation using repeated overlays of several images is a particularly good application for the tools because it depends in no way upon the complexity of the image but strictly on the amount of time required to restore the region to the screen once it is buffered. Even as complex an image as a Mandelbrot set, which may take hours to generate, can be buffered and restored at animation speed if no logical changes need to occur in the image. Drawing the individual frames, buffering each in memory, and displaying them in rapid sequence will produce the flip-card motion effect common in some types of animation, and because you don't need to erase one image before drawing another, the result is nearly smooth and flicker-free.

Saving Screen Regions

Listing One, page 92, contains the Turbo Pascal code for *SaveRegion*, the procedure that saves a screen region into a buffer. You call *SaveRegion* with arguments defining the rectangular screen region to be buffered, expressing the coordinates in absolute screen values; that is, medium-resolution x range 0–319 and y range 0–199 or hi-res x range 0–639 and y range 0–199. In addition, you must declare a pointer referring to a screen buffer containing the saved screen as a variable parameter argument.

SaveRegion itself calls *GETMEM* to allocate sufficient memory to hold the screen buffer, and then it calls *GETPIC* to get the picture from the screen and copy it into this buffer. The details for the *SaveRegion* input and output header are:

```
TYPE buffermemory : ARRAY [1 . . 3]
                              OF INTEGER;
  bufferaddress : ^buffermemory;
PROCEDURE SaveRegion ( VAR buff :
                             bufferaddress;
     x1, y1, {upper left coords}
     x2, y2 {lower right coords}
        : INTEGER );
```

Notice that array of three integers—*buff^[1]*, *buff^[2]*, and *buff^[3]*. Although they are somewhat enigmatically described in the Turbo Pascal reference manual, these three integers describing the parameters of the buffered screen must contain the following values:

- *buff^[1]*—an integer code for the current screen mode, namely 2 for *GRAPHMODE* or *GRAPHCOLORMODE* or 1 for *HIRES* mode
- *buff^[2]*—the width in pixels of the buffered region according to the mode in *buff^[1]*
- *buff^[3]*—the height in pixels of the buffered region

GETPIC stores the remaining screen image data in successive bytes after *buff^[3]*.

The program needs to know how much space the buffer for this screen region will need so it can dynamically allocate the memory via *GETMEM* and inform *GETPIC* about it. The documentation for *GETPIC* in the Turbo Pascal reference manual requires that this size be computed for medium resolution as:

```
size := ((width+3) DIV 4) * height *
                                  2 + 6
```

and for high resolution as:

```
size := ((width+7) DIV 8) * height
                                  + 6
```

These expressions account for the three integers (6 bytes) and the total number of bits needed to represent all pixels, where each pixel is either 2 bits for medium resolution (four colors) or 1 bit for high resolution (two colors). In either case, pixel width and pixel height are computed as follows:

```
width := ABS ( x1–x2 ) + 1;
height := ABS ( y1–y2 ) + 1
```

Calls to *GETMEM* and *GETPIC* with these parameters will allocate contiguous memory dynamically at the location determined by the pointer *buff*:

```
GETMEM ( buff, size );
GETPIC ( buff^, x1, y1, x2, y2 );
```

I use *GETMEM* rather than the standard *NEW* allocation procedure in Pascal because the amount of memory varies at run time for different screen buffer sizes and must be computed. *NEW* allocates space at run time but only according to the static *TYPE* declared at compile time; *GETMEM* permits it to be computed on the fly.

GETPIC actually permits any standard *TYPE* variable to be used to declare the buffer. An integer would suffice in most cases, but I use an array of three integers so that the resolution, width, and height are conveniently accessible using *buff^[1]*, *buff^[2]*, and *buff^[3]*, respectively. You will need these later when you want to display a screen region using *RestoreRegion* and/or free the memory consumed by its buffer.

The number of graphics regions you can save to buffer simultaneously is limited only by heap space.

Note that in Listing One I use the *max* and *min* functions locally within *SaveRegion* to filter the passed coordinates and thereby guarantee that they lie within the specified range for the current resolution mode. Note also the function *CRTmode*, which returns the current resolution mode of the graphics display, thus determining how *size* is computed.

My original versions of *SaveRegion* and *RestoreRegion* didn't have a *CRTmode* function; it required a global variable to carry the currently active resolution. Such globals are a nuisance when designing self-contained tools that abide by loose coupling principles commonly used in well-structured systems. In this case, I found that I could virtually eliminate side effects by creating a function to determine the current video mode. You just load the *AX* register with $0F00 and exercise ROM BIOS interrupt *$10*, and you get back the CRT mode-of-operation parameters in the *AX* and *BX* registers (see AT&T's 6300 documentation). Namely, the low byte of the 16-bit *AX* register contains an integer code corresponding to the current mode of operation (see Listing Two, page 92, for these codes). Although not used here, the high byte of the *AX* register contains the number of character columns in the text display if text mode is active and should be ignored in any graphics mode. The high byte of the *BX* register contains the current display-memory page.

Testing the CRT mode is a simple matter of declaring the usual register variables within a *RECORD* structure, setting the conditions for the interrupt to occur, calling the Turbo Pascal *INTR* procedure, and interpreting the returned results, as follows:

```
FUNCTION CRTmode
  ( VAR char__columns,
    display__page : BYTE ) : BYTE;
TYPE
  regpack = RECORD
    ax,bx,cx,dx,bp,si,di,de,es,flags :IN-
                                 TEGER
END; {regpack}
VAR dosreg : regpack;
BEGIN
  WITH dosreg DO BEGIN
    ax := $0F00;
    INTR ( $10, dosreg );
    CRTmode := LO ( ax );
    char__columns := HI ( ax );
    display__page := HI ( bx )
  END {WITH}
END; {CRTmode}
```

Restoring Regions to the Screen

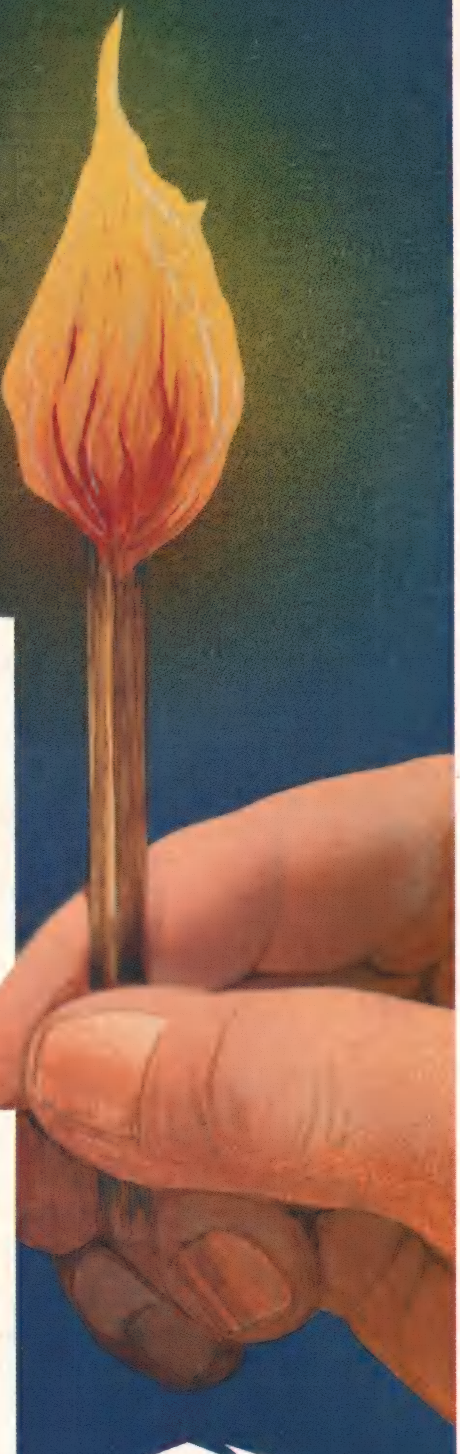
Listing Three, page 93, contains the complete code for the *RestoreRegion* procedure. The heading for the *RestoreRegion* procedure is:

```
PROCEDURE RestoreRegion ( VAR
  buff : bufferaddress; x, y : INTE-
          GER; freeup : BOOLEAN );
```

You call it with arguments describing the lower-left screen coordinates (x,y) where the buffered region is to be placed on the current screen, and it uses the Turbo Pascal-supplied procedure *PUTPIC* to place the image at the point (x,y). *RestoreRegion* also expects the buffer

Unmatched.

If you want unmatched performance and portability, we have it. The hottest file handler and report generator on the market.



The c-tree **file handler** offers unmatched file accessing speed. The r-tree **report generator** makes producing reports a snap. Both packages offer unmatched portability. Thousands of programmers are using these packages in over 50 system environments: DOS, UNIX, XENIX, OS/2, MACINTOSH, VAX, TOWER and .... **YOURS.**

More for your money • complete C-source code • single and multi-user capability • no royalties • unlimited free technical support • port to all machines........for one price.

c-tree features • fixed and variable length data records • record locking • variable length keys and key compression • overcomes OS file limit .... and more.

r-tree features • no printer spacing charts • change reports without recoding • unlimited control breaks, accumulators and virtual field calculations • powerful search, select and sort capabilities over multiple files..... saves days of coding.

FairCom's unmatched products will work for you. Order c-tree today for $395, r-tree for $295. When ordered together, r-tree is only $255. For VISA, MasterCard and C.O.D. orders call (314) 445-6833. For c-tree benchmark comparisons, write us at 4006 West Broadway, Columbia, MO 65203.

CIRCLE 93 ON READER SERVICE CARD

c-tree™ /r-tree™
By FairCom

4006 W. Broadway Columbia, MO 65203

UNIX is trademark of AT&T. MACINTOSH is trademark licensed to Apple Computer Company. VAX is trademark of DEC. TOWER is trademark of NCR. XENIX is trademark of Microsoft

TURBO PASCAL GRAPHICS
(continued from page 39)

pointer used by a previous *SaveRegion* call to be transferred in *buff*. Finally, there's that Boolean argument. I chose to use a *BOOLEAN* (called *freeup*) to indicate either that the restored buffer should be deallocated (*freeup = TRUE*) or that it should not be deallocated (*freeup = FALSE*), implying that the region may be restored to the screen again and again, as with a menu.

One caution: do not use the same buffer name on two successive calls to *SaveRegion* because this will render the first buffer inaccessible and you won't be able to deallocate it. If you must use the same buffer name in this manner, then copy the existing buffer into a *bufferaddress* variable name prior to the second call. Otherwise, a region corresponding to a given named buffer should be restored with the *freeup* argument set to *TRUE* before the same buffer is used again in a *SaveRegion* call.

RestoreRegion uses the resolution, width, and height parameters previously saved in the buffer to compute the amount of memory in *size* to be deallocated by *FREEMEM*. Note that it won't restore the image if any part of the region extends beyond the screen edges (*x_ok* and *y_ok*); two bells will sound in this case. Note, too, that the logic in this procedure permits you to remove this bounding-edge test, permitting partial image restoration.

Figure 1, page 44, illustrates a pop-up graphics menu that is used to overlay a complex fractal image. The fractal image is previously buffered using *SaveRegion*, and when restored using *RestoreRegion*, it overwrites the image containing the menu. The effect in such a case is to erase the menu and restore the background to its previous state.

Managing Memory

I found it wise to include in my bag of tools a procedure called *FreeBuffer* to allow me to deallocate an image buffer without restoring the image visually to the screen. Like *RestoreRegion*, *FreeBuffer* computes the size of the memory block to be freed and calls *FREEMEM* to com-

VEDIT PLUS

#1 PROGRAMMABLE EDITOR

Call 1-800-45-VEDIT for FREE Fully Functional Demo Disk

Stunning speed. Unmatched performance. Total flexibility. Simple and intuitive operation. The newest VEDIT PLUS easily satisfies the most demanding computer professional.

Try a Dazzling Demo Yourself.

The free demo disk is fully functional—you can try all features yourself. Best, the demo includes a dazzling menu-driven tutorial—you experiment in one window while another gives instructions.

The powerful "macro" programming language helps you eliminate repetitive editing tasks. The impressive demo/tutorial is written entirely as a "macro"—it shows that no other editor's "macro" language even comes close. And VEDIT PLUS is only 40K in size.

Go ahead. Call for your free demo today, You'll see why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

Only VEDIT PLUS is this Flexible.

The installation lets you pick from closely emulating the keyboard layout of Word Perfect, WordStar and others. Or you can easily create your own layout and even your own editing functions. Supports any screen size—you pick screen colors and attributes.

Supports the IBM PC, XT, AT and PS/2. Also supports MultiLink, PC-MOS/386, Concurrent DOS and most networks. Also available for MS-DOS, FlexOS (protected mode), CP/M-86 and CP/M. (Yes, we support windows on most CRT terminals, including CRTs connected to an IBM PC.) Order direct or from your dealer. $185.

Special: VEDIT (single file, no windows) for CP/M—$49.

- **Fully Network Compatible**
- **Call for XENIX-286 version**
- **30 Day Money-back guarantee**

Compare Features and Speed

	BRIEF	Norton Editor	PMATE	VEDIT PLUS
'Off the cuff' macros	No	No	Yes	Yes
Built-in macros	Yes	No	Yes	Yes
Keystroke macros	Only 1	No	No	Unlimited
Multiple file editing	20 +	2	No	20 +
Windows	20 +	2	No	20 +
Macro execution window	No	No	No	Yes
Pop-up menus	No	No	No	Yes
Execute DOS commands	Yes	Yes	Yes	Yes
Automatic processing of Compiler errors	Yes	No	No	Yes
"Cut and paste" buffers	1	1	1	36
Undo line changes	Yes	No	No	Yes
Paragraph justification	No	No	No	Yes
Convert to/from WordStar	No	No	No	Yes
On-line calculator	No	No	No	Yes
Configurable Keyboard	Hard	No	Hard	Easy
43 line EGA support	Yes	No	No	Yes
Manual size/index	250/No	42/no	469/Yes	380/Yes
Benchmarks in 120K File:				
2000 replacements	1:15 min	34 sec	1:07 min	6 sec
Pattern matching search	20 sec	Cannot	Cannot	2 sec
Pattern matching replace	2:40 min	Cannot	Cannot	11 sec

VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PMATE is a trademark of Phoenix Technologies Ltd. Norton Editor is a trademark of Peter Norton Computing Inc. MultiLink and PC-MOS/386 are trademarks of the The Software Link, Inc. CP/M and FlexOS are trademarks of Digital Research. MS-DOS is a trademark of Microsoft.

*Also available for TI Professional, Tandy 2000, DEC Rainbow, Wyse WY700 and others.
*Demo disk is fully functional, but does not readily write large files.

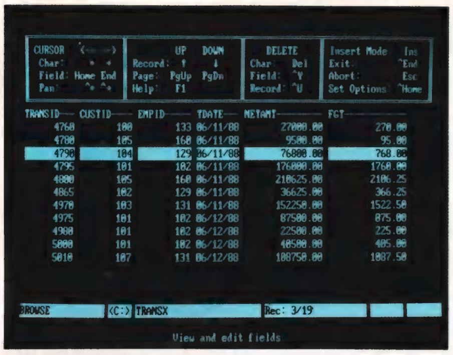
CIRCLE 122 ON READER SERVICE CARD

CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103
(313) 996-1299, TELEX 701821

OUR NEW GRAPHICS SOFTW
MAY BE A BIT FASTER TH

ARE FOR dBASE AND R:BASE AN WHAT YOU'RE USED TO.



```
CURSOR   <-- -->  |         UP     DOWN  |   DELETE       | Insert Mode    Ins
 Char:   <-  ->   | Record:  ↑     ↓     | Char:    Del   | Exit:          ^End
 Field: Home End  | Page:   PgUp  PgDn   | Field:   ^Y    | Abort:         Esc
 Pan:    ^<- ^->  | Help:   F1           | Record:  ^U    | Set Options:   ^Home

TRANSID--- CUSTID---- EMPID------ TDATE--- NETAMT--------- FGT-----------
     4760        100          133 06/11/88        27000.00          270.00
     4780        105          160 06/11/88         9500.00           95.00
     4790        104          129 06/11/88        76800.00          768.00
     4795        101          102 06/11/88       176000.00         1760.00
     4800        105          160 06/11/88       210625.00         2106.25
     4865        102          129 06/11/88        36625.00          366.25
     4970        103          131 06/11/88       152250.00         1522.50
     4975        101          102 06/12/88        87500.00          875.00
     4980        101          102 06/12/88        22500.00          225.00
     5000        101          102 06/12/88        40500.00          405.00
     5010        107          131 06/12/88       108750.00         1087.50

BROWSE          <C:> TRANSX                 Rec: 3/19

                          View and edit fields
```

DB Graphics turns raw dBASE III/III PLUS and R:BASE data files into graphics with no conversion, export or import.

Up until now, turning the raw data of a dBASE III/III PLUS or R:BASE file into easily understood graphs and charts has been anything but quick.

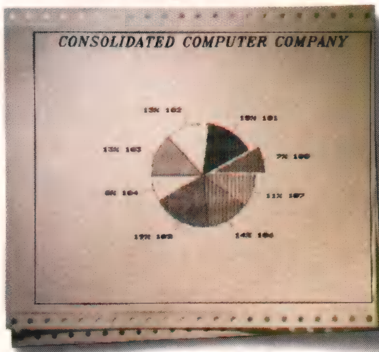
It's meant plodding back and forth between your database and your graphics program, through file conversions and translations, until you finally got a chart that made your point.

But now there's a faster route to arresting presentation graphics. Now there's DB Graphics.

THE FIRST GRAPHICS SOFTWARE THAT GOES STRAIGHT TO THE HEART OF YOUR DATABASE.

DB Graphics is the only presentation graphics software designed specifically for dBASE, R:BASE and other database software users.

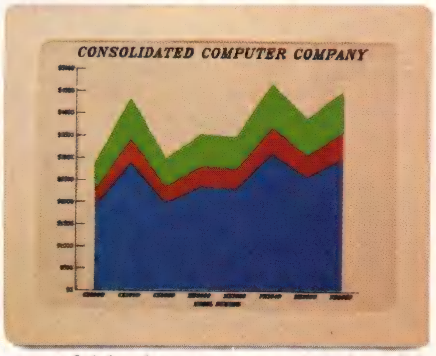
Since it reads dBASE and R:BASE databases directly, there are no time-consuming export, import or conversion utilities to fool with. DB Graphics can also sort and group fields and use conditional operators and other powerful database management tools to precisely select and graphically display the data you need. And because it has a direct line to your live data, you can modify a graph, or choose different data to represent, in an instant.



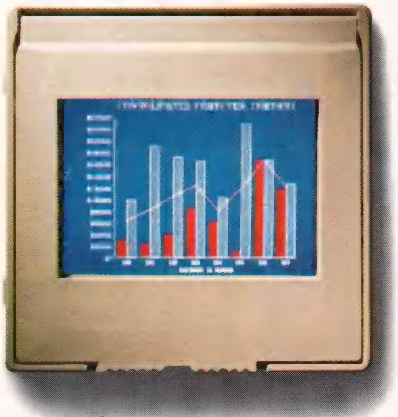
You can change the style of a graphic, or the data it represents in an instant.

MAKE A GRAPHIC IMPROVEMENT IN ALL YOUR PRESENTATIONS.

Even though DB Graphics is more intelligent than other graphics programs, it's no less attractive. It allows you to display your data in eight different graph types—pie, bar, high/low, area, scatter, mixed, column, and line. You can work with 16 colors and hundreds of vivid combinations of textures and patterns. Plus, you can choose from seven type styles for free text and labels.



Powerful database management tools make it easy to get the data you need into a graph.

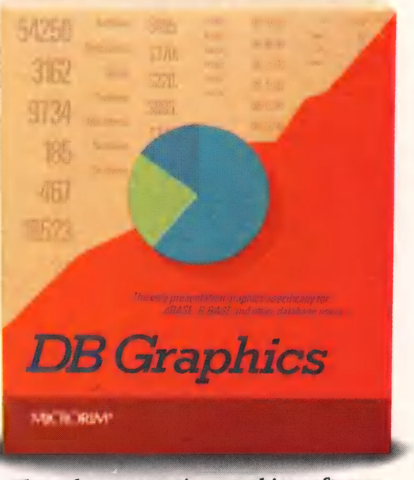

DB Graphics can produce a variety of output, including 35mm slides.

In fact, there's really only one kind of database presentation DB Graphics isn't capable of making. A dull one.

FOR A TRIAL PACK, CALL 1-800-624-0810 DEPT. DR1187.

We could go on about the power, speed and accuracy of DB Graphics, but you really need to experience it for yourself.

So call our toll-free number (or from Alaska or Canada call 1-206-867-1800 Dept. DR1187) and order a DB Graphics Trial Pack for just $9.95 today. And start making the most out of your database.



The only presentation graphics software specifically designed for dBASE, R:BASE and other database software users.

DB Graphics also works with Lotus 1-2-3, Symphony, Multiplan and other data management files after translating to the ASCII, dBASE III/III PLUS or R:BASE format. DB Graphics is compatible with the IBM PS/2, PC, XT, AT and 100% compatible personal computers. It supports the IBM VGA, MCGA, EGA, CGA and Hercules display standards. It also supports most popular plotters, graphic printers, camera systems and laser printers from IBM, Epson, Hewlett-Packard and many others. © 1987 Microrim.

CIRCLE 320 ON READER SERVICE CARD

(continued on page 40)

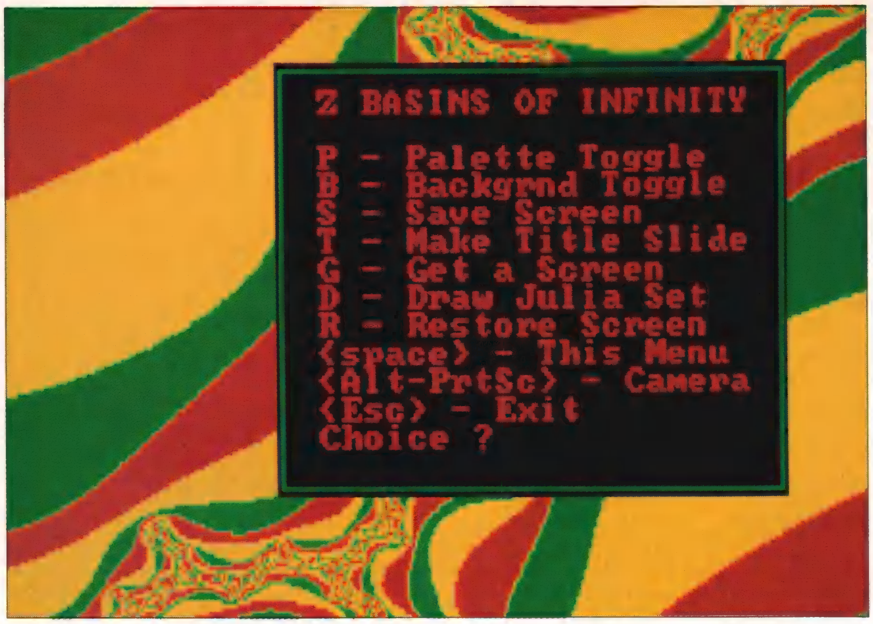


Figure 1: *The pop-up menu, previously drawn and saved, overlaying the fractal image*

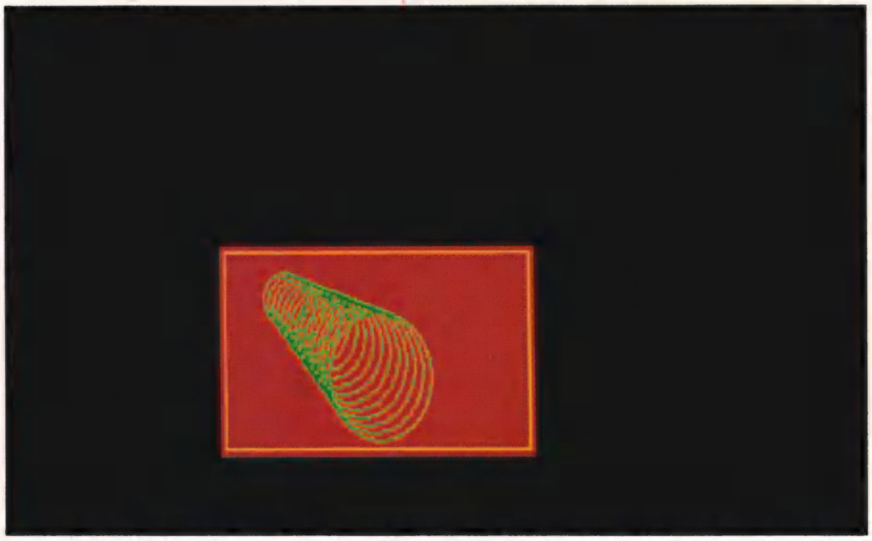


Figure 2: *A saved graphic region used in Listing Six as* buffer

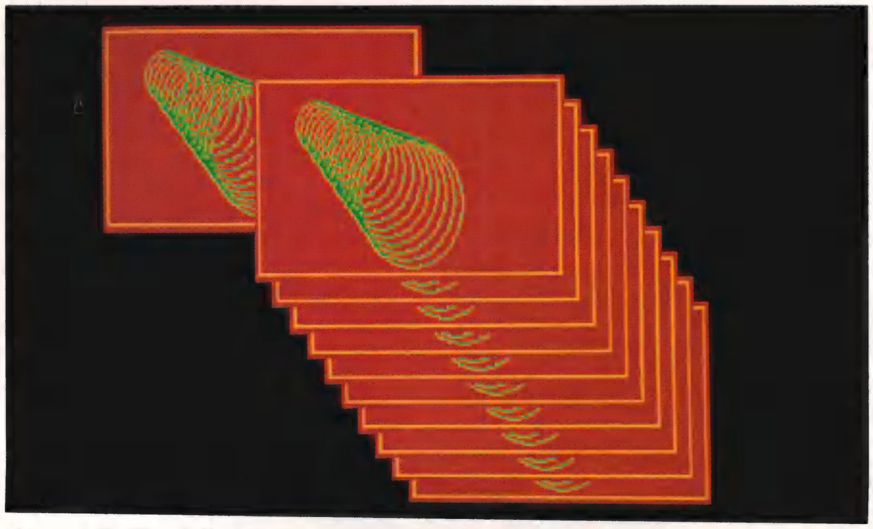


Figure 3: *The full screen saved as* buffer2 *in Listing Six*

plete the job. Listing Four, page 94, contains the complete *FreeBuffer* procedure. The procedure heading takes this form:

```
PROCEDURE FreeBuffer
  ( VAR buff : bufferaddress );
```

Storing Regions on Disk

Because an image is stored within a contiguous section of memory, it is possible to save it rapidly to an "untyped" disk file and later load it from disk into a buffer. Turbo Pascal provides for such untyped files, and its *BLOCKWRITE* and *BLOCKREAD*, as described on page 114 of the reference manual, will suffice for writing and reading the images.

You call *BLOCKWRITE* like this:

```
numrecs := 1 + (size–1) DIV 128;
BLOCKWRITE ( FileVariable, buff^,
                          numrecs )
```

because it saves a block of memory to a file named *FileVariable* consisting of *numrecs* 128-byte block records starting at *buff*. The last record may conceivably contain less than 128 bytes of true image data, wasting a few bytes, but the sacrifice of a few bytes of memory and disk space is well worth what you gain in speed by using block disk transfers.

To load an image back from disk into a buffer, you determine the size of the image file (in 128-byte records) using Turbo Pascal's *FILESIZE* function, allocate sufficient memory to hold the image using *GETMEM*, and finally, *BLOCKREAD* the records from the original "block written" file into the buffer:

```
SizeOfFile := FILESIZE ( FileVari-
                              able );
GETMEM ( buff, SizeOfFile * 128 );
BLOCKREAD ( FileVariable, buff^,
                       SizeOfFile );
```

Listing Five, page 94, contains the simple procedures *SaveBlockToDisk* and *GetBlockFromDisk* for saving and loading these buffered regions to and from disk.

Examples

Having hopefully whetted your appetite to see a demo, I submit the demo program in Listing Six, page 95. It creates the graphic region shown in Figure 2, page 44, saves it in a buffer (*buffer*), and restores it to various locations on the screen. A second buffer, *buffer2*, contains a full-screen image (see Figure 3, page 44) that serves as background for subsequent pop-up overlays. The program restores this image alternately with the smaller overlay, creating a motion effect.

Note that I have included GRAPH.P Extended Graphics (supplied with Turbo Pascal, Version 3.0) as an option if you want to use windows via *GRAPHWINDOW* and the window-clearing operation *FILLSCREEN*. If you use Turbo Pascal windows in this manner, then the coordinates in *SaveRegion* and *RestoreRegion* refer to the current window, where (0,0) is its upper-left corner.

Availability

All the source code for articles in this issue is available on a single disk. To order, send $14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call (415) 366-3600, ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

Bibliography

System Programmer's Guide. AT&T 6300 Personal Computer Documentation, 1986.

Turbo Pascal V3.0 Reference Manual. Scotts Valley, Calif.: Borland International, 1987.

DDJ

(Listings begin on page 92.)

Vote for your favorite feature/article. Circle Reader Service **No. 3.**

Brand New From Peter Norton
A PROGRAMMER'S EDITOR

that's *lightning fast* with the *hot* features programmers need

only $50

THE NORTON EDITOR™

Direct from the man who gave you *The Norton Utilities, Inside the IBM PC,* and the *Peter Norton Programmer's Guide.*

"This is the programmer's editor that I wished I'd had when I wrote my *Norton Utilities*. You can ***program your way to glory*** with *The Norton Editor.*"

Peter Norton

Easily customized, and saved
Split-screen editing
A wonderful condensed/outline display
Great for assembler, Pascal *and* C

Peter Norton Computing, Inc., 2210 Wilshire Boulevard, Santa Monica, CA 90403, 213-453-2361. Visa, Mastercard and phone orders welcome.

The Norton Editor™ is a trademark of Peter Norton Computing, Inc. © 1986 Peter Norton Computing

CIRCLE 243 ON READER SERVICE CARD

UNIX/C WINDOW DEVELOPMENT COMPATIBILITY with **CURSES** for MS-DOS and MS-OS/2.

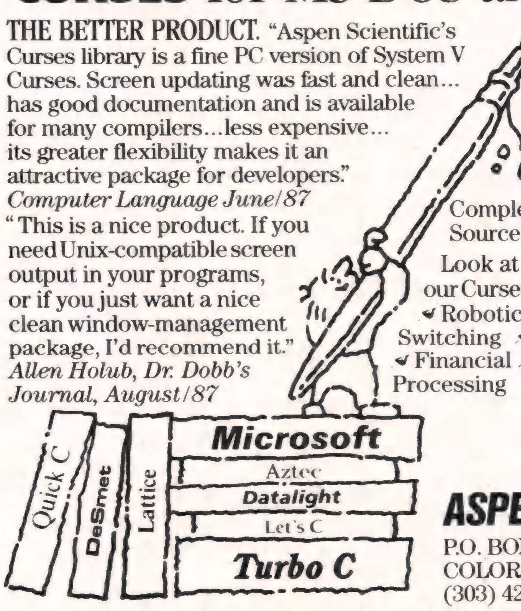
THE BETTER PRODUCT. "Aspen Scientific's Curses library is a fine PC version of System V Curses. Screen updating was fast and clean... has good documentation and is available for many compilers...less expensive... its greater flexibility makes it an attractive package for developers."
Computer Language June/87

"This is a nice product. If you need Unix-compatible screen output in your programs, or if you just want a nice clean window-management package, I'd recommend it."
Allen Holub, Dr. Dobb's Journal, August/87

FREE-FAST UNIX compatible forms tool kit with source code. Included if you *ORDER RIGHT NOW.*

Complete curses tool kit: $119.
Source code available for: $289.

Look at a few of the applications our Curses is benefiting: ✓ Aerospace ✓ Robotics ✓ Consulting ✓ Telephone Switching ✓ AI ✓ Voice Recognition ✓ Financial ✓ Engineering ✓ Word Processing



ASPEN SCIENTIFIC
P.O. BOX 72 WHEAT RIDGE, COLORADO 80034-0072
(303) 423-8088

Trademarks: MS-DOS, MS-Windows, MS-OS/2, XENIX, Quick C (Microsoft); Unix (AT&T Bell Labs); Turbo C (Borland); Aztec (Manx); Lattice (Lattice); Let's C (Mark Williams); DeSmet (DeSmet Software); Datalight (Datalight).

CIRCLE 360 ON READER SERVICE CARD

Using EGA Graphics Screens in Your Programs

by J. Brooks Breeden

Getting a full-color screen from a paint program into EGA display memory from a high-level language

Creating detailed images for the Enhanced Graphics Adapter (EGA) usually means graph paper, binary-to-hexadecimal conversion, and counting lots of dots. Commercial "paint" programs such as EGAPaint and PC Paintbrush, on the other hand, offer real-time, see-what-you-draw image creation but typically only allow full screens of images to be chained in a sort of "slide show."

Once you display a screen (created by such a paint program) with your programming language, you can then use your language's graphics commands to modify the screen. You can capture bit blocks, animate portions of the screen, and save images to disk—all this and more, with no graph paper or binary-to-hexadecimal conversion and only a little pixel counting.

This article presents Forth routines that can be used to load an EGAPaint file into video memory for producing animation (see Listing One, page 88). Although Forth is the language used in the listing, the pseudocode in Listing Two, page 88, should enable you to adapt the method to any language supporting EGA graphics. The process is relatively straightforward. Listing Three, page 89, contains a demonstration of the routines' use in a simple game. But first, let's take a brief look at the Enhanced Graphics Adapter.

J. Brooks Breeden, 367 Brynhild Rd., Columbus, OH 43202. Brooks is a professor at Ohio State University. He worked with CAI on mainframes from 1976 until 1981, when he switched to using the IBM PC. He has worked with Forth exclusively since 1984.

An EGA Review

Since its introduction in 1984, the IBM Enhanced Graphics Adapter (EGA) has received much less press than did its older sibling, the Color Graphics Adapter (CGA), in its first three years. The CGA's introduction was followed by a plethora of $20 books on how to program graphics (mostly in BASIC), but despite the introduction of many low-price EGA clone boards, information on taking advantage of the EGA's capabilities is still scarce. The EGA is more complex than the CGA and programming the EGA does require some understanding of the fundamental differences between them.

In its high-resolution mode (640 × 200 pixels, black and white), the CGA uses 1 byte to store data for eight pixels, 1 bit per pixel, either on or off (see Figure 1, page 47). The CGA memory required for a single page of hi-res display is therefore 16,000 bytes (640 × 200/8 = 16,000), although 16K (16 × 1,024 = 16,384 bytes) is allocated. CGA memory begins at B800h for even scan lines; odd scan lines are offset by 8K (8,192 bytes). The extra 192 bytes are not used in either area.

In contrast, EGA hi-res graphics display memory begins at A000h (see Figure 2, page 47). Unlike the CGA, display memory is continuous, not separated into even-line and odd-line locations. In its high-resolution 16-color mode, the EGA uses 1 byte to represent 16-color data for eight pixels, 1 bit per pixel. No, you didn't read that wrong.

Forgetting color for the moment, the EGA display is a matrix of 224,000 dots (640 × 350 = 224,000). If the display is either black or white, each dot or pixel can be represented by 1 bit (either on or off), just like the CGA in its hi-res mode. One byte represents eight contiguous pixels on a scan line, so 28,000 bytes (224,000/8 = 28,000) contain pixel data for one full screen.

But what about color? Let's first examine how a color printer "paints" an image. Color printers usually make four passes per line. Simply stated, color printing involves mapping areas of a page that receive a color, then applying the ink to those areas of the page. Successive mapping and application of cyan, magenta, yellow, and black inks (pigments that reflect light) yield a full-color image.

The EGA similarly uses a "map mask" to determine which phosphors (that emit light) should be excited in red, green, blue, and intensity "planes." One screen full of pixels (28,000 bytes) maps all areas of the screen that contain red in the displayed color, a second 28,000-byte screen maps all areas containing blue, a third maps green, and a fourth maps the associated intensity. The pixels mapped from all four planes blend visually to make the full-range color display. The EGA display, then, can be visualized as four overlaid planes of color, numbered 0–3, each having the same address—A000h. Yes, the *same* address. If the CGA is a single family residence, the EGA is a four-unit apartment building.

In addition to display memory, the EGA card contains control registers and temporary latches. The control registers are accessed by writing data to a port. To write to the display, you first set the control registers and read the byte at the desired address and then write the new byte to the same address. Reading the byte (eight pixels at a time) causes the EGA to read 4 bytes, one from each plane, into the temporary latches (eight pixels' worth of data in four colors). When a data byte is then written back to the same address, the values in the control registers at that time determine how the data gets written to each of the four planes. Map masks determine which plane is written to, and bit masks determine which bits are on in the byte that is written to the plane. The bibliography lists references that cover this sort of low-level programming in detail.

High-level languages supporting the EGA typically hide the complexity of such bit-level operations in commands such as *LINE*, *ARC*, and so on, but most do not implement all of the EGA functions. To do some things, you still have to resort to low-level programming. Although map-mask/bit-mask programming may appear excessively tedious, it is easy to do some things with the EGA, and moving data from storage to the display is one of the easiest. Displaying a screen image on the EGA means simply moving data representing the blue, red, green, and intensity planes from the source to the display address (the same address, the video *segment:offset*). You set the control registers (by writing values to the ports) such that the latches send the data being read to the appropriate plane. (Sending the blue data to the red plane results in bizarre color schemes. Go ahead, try it!) The only problem is finding out where the data for each plane is stored in your paint program's file.

Paint Your Wagon

EGAPaint, a popular paint program from RIX Softworks is used in the listings. Like other paint programs, it allows screens of images created with a program to be printed or saved to disks as files (either compressed or uncompressed). A printer utility, EGAPrint, is included to allow capture of anything being displayed to an uncompressed file in EGAPaint format.

The standard 640 × 350-pixel, 16-color file format for EGAPaint 2001 is 112,016 bytes: 16 bytes of color palette information, followed by 28,000 bytes each of blue, red, green, and intensity, in that order. (I called RIX and asked!)

The newer EGAPaint 2005 file formats vary depending upon the type of screen—for example, 640 × 350, 640 × 480, and so on. RIX says it will share uncompressed file-format information with any registered owner of the program who writes and requests it (I'm still waiting), but it will not share the compression scheme it uses. *ARC* seems to work fine for long-term storage, though. Other paint programs may save color plane data in a different order and might include palette color data in a different location (or not at all). A letter or call to the program vendor is worthwhile if some experimentation doesn't discover the proper sequence quickly.

Moving Right Along

You can program the EGA using any language that lets you do the following things:

1. read and write to an absolute address
2. read or write to a specific I/O port
3. call an external subroutine or a DOS interrupt

For example, to set write mode 0 in Turbo Pascal you use *port[$03CE]:=5* to select the register, then *port[$03CF]:=0* to set the register value. In BASIC you use *out*:

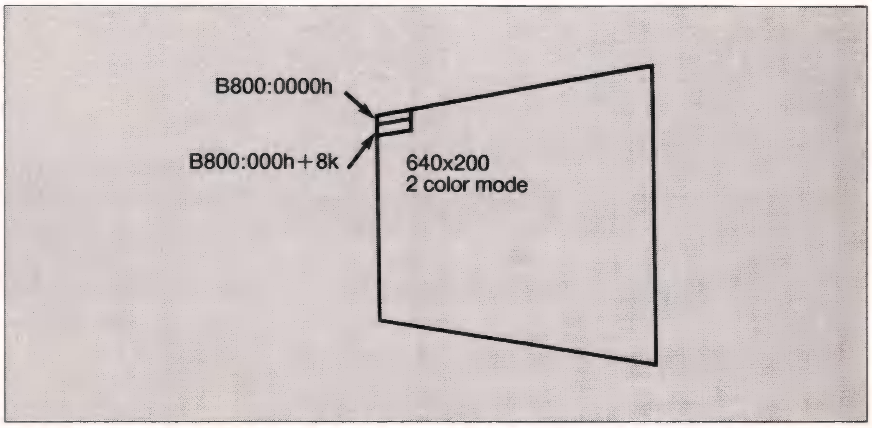


Figure 1: *Conceptual representation of CGA display memory. The plane is 16,000 bytes; 1 byte controls eight pixels.*

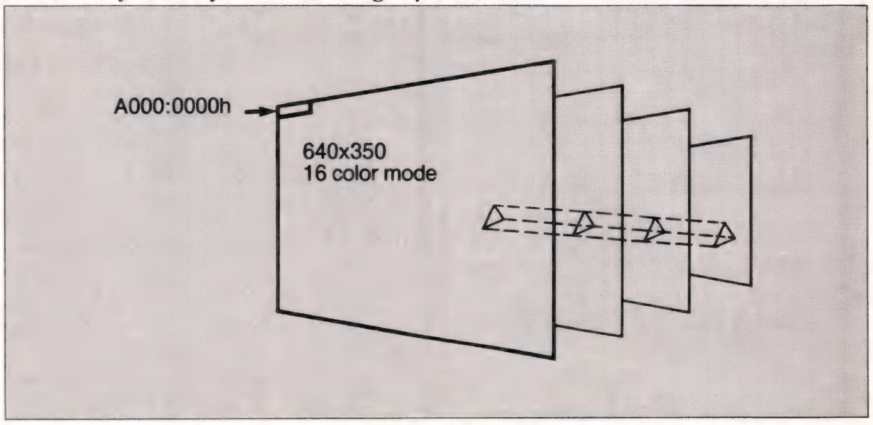


Figure 2: *Conceptual representation of EGA display memory. Each plane is 28,000 bytes; 1 byte controls eight pixels for all planes. You read and write to the same address. The latches deliver to the previously selected plane(s).*

(continued from page 47)

```
100 out &h3ce,5
110 out &h3cf,&h0
```

In UR/FORTH, the word *PC!* sends a byte to a port. The sequence becomes a word:

```
: WriteMode0 ( --- )
   HEX 5 3CE PC! 0 3CF PC! DECIMAL ;
```

Listing One contains the required routines written in UR/FORTH 1.01 (MS-DOS version), a Forth-83 implementation from Laboratory Microsystems; Listing Two contains a pseudocode version. The listings don't include saving a screen to disk because the paint program creates the image and does the save-to-disk. As they say in academe, reversing the process to save a screen to disk is left as an exercise . . . .

A Mindless Game of Motor Skill

Figure 3, page 49, is a dump from my sample program. EGAPaint was used to create a rear view of a Fokker Dr.1 triplane. After 8 × Zoom "detailing," the Fokker was saved as an EGAPaint file of 112,016 bytes. A UR/FORTH application then loaded the EGAPaint image using the routines shown in Listing One. Once the EGAPaint screen was displayed, UR/FORTH's bit-block save routine, *@BLOCK*, was used to save a rectangle of screen image surrounding the Fokker to memory. (That's where the pixel-counting comes in.) The block image was then written to disk by UR/FORTH as a DOS file of approximately 3K, and the 112,016 byte EGAPaint file was deleted.

This Fokker image was used as the principle image for a World War I shoot-'em-up dogfight. In the game, the Fokker image is loaded from disk to an allotted memory area and repeatedly written to the display using UR/FORTH's *!BLOCK*. Obviously, cursor keys control (move the Fokker relative to) the gunsight of your aircraft, and the Fokker pilot has a will (mind?) of his own as his range of random motion changes as hits are scored.

Animation of detailed images is a natural outgrowth of this technique because complicated rotations and so on can be captured as a series of frames that can be loaded into memory, rapidly swapped, then discarded. Currently, I'm using this method in developing computer-assisted instruction (CAI) modules for my courses. As virtual memory becomes commonplace with OS/2 et al., we may see this technique used increasingly. Until then, experiment, and if you discover something, for goodness sake, publish it!

The program begins with initial credits and a menu. When play begins, a Fokker Dr.1 triplane, a gunsight, your remaining ammunition, and the number of hits scored are displayed. The Fokker is moving slowly and is unaware of your presence. Your mission is to maneuver your plane (the gunsight) into position behind the Fokker and score 20 hits in the fuselage area. The space bar fires the guns.

Remember, you are flying the pursuit aircraft (the gunsight) not the Fokker, so the controls take a little getting used to. The number pad is your "stick." Pressing the up arrow

Create 68K Embedded Systems On ATs

Oregon Software Brings VAX and VME Pascal-2® Cross-Development Tools To MS-DOS Workstations

Get All Your Tools In One Package

Compiler
- 68000, 68020, and 68881 instruction sets
- ROMable code

Assembler-Linker
Concurrent Program Support
Stand-Alone Library
Source Management

OREGON SOFTWARE

800-367-2202
6915 SW Macadam Avenue, Portland, Oregon 97219 U.S.A.
503-245-2202 FAX 503-245-8449 TWX 910-464-4779

SCREEN GENERATOR - WINDOW MANAGER

POPSCREEN™

Programmers - Use The Best!

Powerful But Simple Screen Editor
Makes easy work of creating complex & colorful displays, pop-up menus, windows: easy boxes, outlines, block moves, superimpose, complete color & character graphics control, boilerplating, on line help; & much more.

Displays Integrate With Your Compiler's Output
PopScreen outputs your displays to library modules for your compiler. Displays link into your program at link time. No frustrating resident loaders, no loading screens from files on disk. Your displays are part of your program!

Instant Screen Access
Your displays pop to the screen instantaneously - in 1/25th of a second! Open and close windows with a simple procedure call. Your displays are fast, professional, and optimally compressed for storage inside your program.

PopScreen 3.0 Supports:
C: Microsoft, Lattice, Turbo, IBM
8086: will output assembly code
PASCAL: IBM, Microsoft
TURBO PASCAL: (inline code)
QUICKBASIC: library modules
DBASE III+ (loadable .bin files)

PopScreen only **$39.95**
If it's not the best, send it back for a full refund!

BaySoft Box 6562-D, Albany, Cal. 94706
415-527-3300

60 DAY SATISFACTION GUARANTEE

key (8) pushes the stick forward, and because the gunsight is "fixed" in the center of the screen, diving makes the Fokker move (relatively) up! Similarly, pulling back on the stick with the down arrow key, pulls your plane's nose up, making the Fokker move down. Left and right arrow keys move you left (the Fokker moves right) and right (the Fokker moves left). Got it? The diagonal keys function, also. This is useful because you will often want to pull up and left, or push down and right, simultaneously.

When no cursor key is pressed, the Fokker simply flies off-screen upward to the left, which is analogous to your plane diving to the right. This is undoubtedly a function of the random-number generator's less-than-perfect randomness. What is ironic, though, is that it is exactly what would tend to happen were you to let go of the controls in a Sopwith Camel. "The Camel spun very quickly, had a very sensitive elevator control, and was very quick on right-hand turns due to the gyroscopic effects of the heavy rotary-engine and the short fuselage."

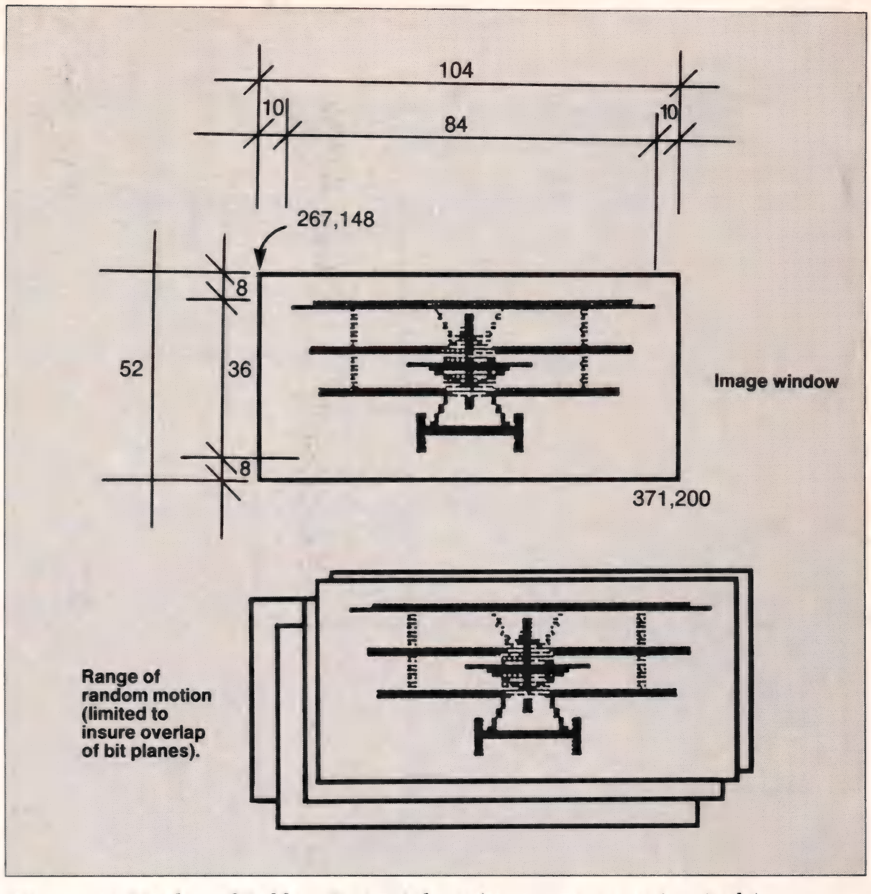
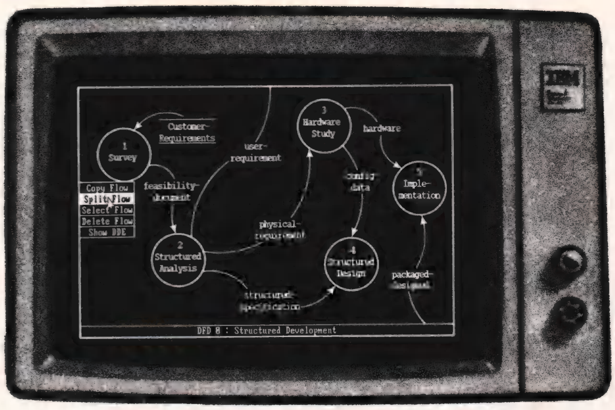

104
10
84
10
267,148
8
52
36
8
Image window
371,200
Range of random motion (limited to insure overlap of bit planes).

Figure 3: *Display of Fokker Dr.1 triplane (measurements in pixels)*

Structured Analysis breakthrough!

THE FIRST **COMPLETE** SA SOFTWARE FOR UNDER $1,000.

Discover the power of computer-aided Structured Analysis...Create specifications more efficiently, more accurately,...With **Teamwork/PCSA**,™ a complete, automated SA tool for your PC **for only $995.**



No other system includes these features for under $1,000:
- Full support of Yourdon/DeMarco SA techniques
- Easy-to-use mouse driven interface with pop-up menus
- Includes integrated project data dictionary
- Includes consistency checking and diagram balancing
- **Now also includes P-Specs, Postscript™ output, and new, easy tutorial**

Teamwork/PCSA runs on most IBM®—compatible PCs. It's used by leading developers at Boeing, AT&T, GE, HP, and Bank of America. And it's the only PC-based software that offers you a growth path to the Teamwork family of CASE tools for real-time modeling, system design and life-cycle management.

CADRE
Cadre Technologies Inc.
222 Richmond St.
Providence, RI 02903

IBM is a registered trademark of International Business Machines. Postscript is a registered trademark of Adobe Systems, Inc.

DDJ 11/87

FREE DeMarco Book! We'll give you two reasons to order **Teamwork/PCSA** today. ONE: you get a 30-day money-back guarantee, so there's absolutely no risk. TWO: Order now and we'll send you **Structured Analysis and System Specification** by Tom DeMarco. It's the "Bible" of structured analysis and normally sells for over $40. And it's yours free. For details or to place your order, call or write today.

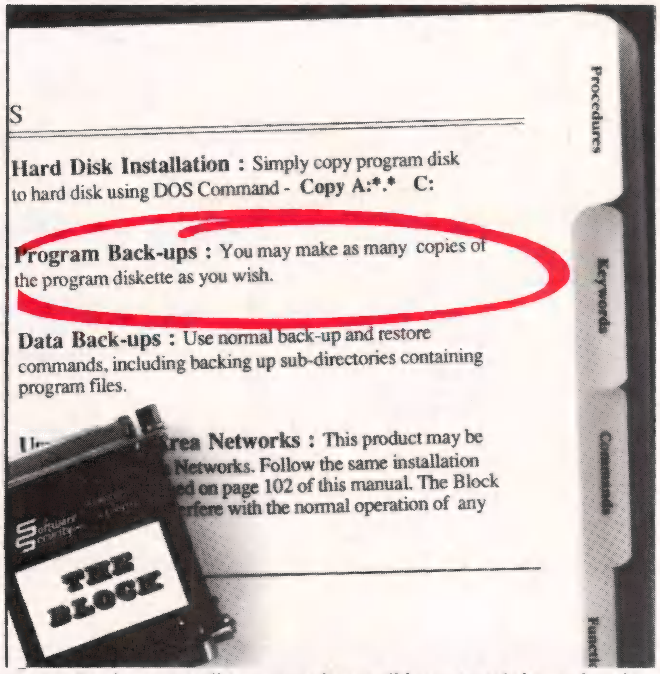
CALL (401) 351-CASE.

North American prices only. Volume discounts available.

CIRCLE 261 ON READER SERVICE CARD

"How to protect your software by letting people copy it."

By Dick Erett, President of Software Security


S

Hard Disk Installation : Simply copy program disk to hard disk using DOS Command - **Copy A:*.* C:**

Program Back-ups : You may make as many copies of the program diskette as you wish.

Data Back-ups : Use normal back-up and restore commands, including backing up sub-directories containing program files.

U[illegible] rea Networks : This product may be [illegible] Networks. Follow the same installation [illegible] ed on page 102 of this manual. The Block [illegible] erfere with the normal operation of any

Soon all software installation procedures will be as straightforward as this. The only difference will be whether you include the option to steal your product or not.

Inventor and entrepreneur, Dick Erett, explains his company's view on the protection of intellectual property.

"A crucial point that even sophisticated software development companies and the trade press seem to be missing or ignoring is this:

Software protection must be understood to be a distinctively different concept from that commonly referred to as copy protection.

Fundamentally, software protection involves devising a method that prevents unauthorized use of a program, without restricting a legitimate user from making any number of additional copies or preventing program operation via hard disk or LANs.

Logic dictates that magnetic media can no more protect itself from misuse than a padlock can lock itself.

Software protection must reside outside the actual storage media. The technique can then be made as tamper proof as deemed necessary. If one is clever enough, patent law can be brought to bear on the method.

Software protection is at a crossroads and the choices are clear. You can give product away to a segment of the market, or take a stand against the theft of your intellectual property.

"...giving your software away is fine..."

We strongly believe that giving your software away is fine, if you make the decision to do so. However, if the public's sense of ethics is determining company policy, then you are no longer in control.

We have patented a device that protects your software while allowing unlimited archival copies and uninhibited use of hard disks and LANs. The name of this product is The BLOCK™.

The BLOCK is the only patented method we know of to protect your investment. It answers all the complaints of reasonable people concerning software protection.

In reality, the only people who could object are those who would like the option of stealing your company's product.

"...eliminating the rationale for copy-busting..."

Since The BLOCK allows a user to make unlimited archival copies the rationale for copy-busting programs is eliminated.

The BLOCK is fully protected by federal patent law rather than the less effective copyright statutes. The law clearly prohibits the production of work-alike devices to replace The BLOCK.

The BLOCK attaches to any communications port of virtually any microcomputer. It comes with a unique customer product number programmed into the circuit.

The BLOCK is transparent to any device attached to the port. Once it is in place users are essentially unaware of its presence. The BLOCK may be daisy-chained to provide security for more than one software package.

Each software developer devises their own procedure for accessing The BLOCK to confirm a legitimate user. If it is not present, then the program can take appropriate action.

"...possibilities... limited only by your imagination..."

The elegance of The BLOCK lies in its simplicity. Once you understand the principle of The BLOCK, hundreds of possibilities will manifest themselves, limited only by your imagination.

Your efforts, investments and intellectual property belong to you, and you have an obligation to protect them. Let us help you safeguard what's rightfully yours. Call today for our brochure, or a demo unit."

Software Security inc.

870 High Ridge Road Stamford, Connecticut 06905

203 329 8870

CIRCLE 170 ON READER SERVICE CARD

EGA GRAPHICS
(continued from page 49)

(Campbell, 1984).

To just type and run the program without modification, you need UR/FORTH (a segmented Forth-83 model) from Laboratory Microsystems. The video driver, EGAGRAPH.EXE, must be installed before UR/FORTH. The file DERFOKKR.IMG, which contains the image created with EGAPaint, must also be in the active directory. Using the technique I have already described, you may wish to build your own aircraft with a paint program, load it into video RAM, and use the bit-block operator *@BLOCK* to capture the image to memory. You then only have to write the image to a file from which it can be loaded whenever needed.

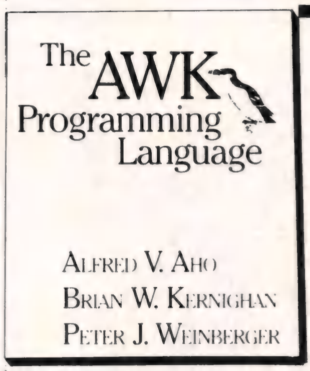
The fundamentals of the game are neither specific to Forth nor to the EGA. Alternately, you can build an aircraft using simple lines and boxes that will work adequately with the CGA. Figure 3 shows the basis for the bit map and how that image is overwritten. Because the Fokker bit map is located by the coordinates of the upper-left corner, the fuselage "hit zone" is simply offset by a similar range of coordinates surrounding the corner.

The area of sky within the bit map beyond the actual image is used to blot parts of the previous image when a new one is written. The extent of the "extra" sky is thus directly related to the amount of random motion allowed with the image. Experimentation resulted in limiting the maximum range of random movement from the current location to plus or minus four pixels. Combined with a possible two additional pixels' movement from a cursor keypress, the maximum Fokker movement is six pixels per loop cycle. The area of sky surrounding the Fokker would allow for more random motion without leaving garbage all over the screen, but I don't think you'll really want it.

The program was developed using both a Compaq Deskpro (8086) and a 6-MHz IBM PC/AT, with the QUICKEYS.COM keyboard speedup program. QUICKEYS.COM was listed in a back issue of *PC Magazine* and can be downloaded from its IRS bulletin

AWK
The Software
MKS AwK

- the **only** complete version of AWK available for DOS;
- **fully compatible** with the latest description in **The AWK Programming Language**, by Aho, Weinberger, and Kernighan;
- easy to learn, giving beginners results with little effort;
- the natural introduction to mastering the C programming language;
- text substitution and pattern matching;
- definable functions;
- associative arrays and regular expressions;
- hardware floating point and large model version;
- rapid prototyping tool for larger programs.


The AWK Programming Language

Alfred V. Aho
Brian W. Kernighan
Peter J. Weinberger

The Book

- written by the authors of the original UNIX-based program, Alfred V. Aho, Peter J. Weinberger, and Brian W. Kernighan;
- the definitive book on AWK just as "The C Programming Language," by Kernighan and Ritchie, defined C;
- recently published by Addison-Wesley;
- lists at $21.95;
- with MKS AWK only $14.00.

Experience the power of a UNIX programming language on your desktop PC without sacrificing your investment in DOS applications and training.

AWK is a versatile first language for non-programmers and a sophisticated data retrieval and report generation tool for the experienced user. Based on a sequence of terse pattern/action rules, AWK allows you to manipulate files for retrieval, transformation, reduction, and validation of data. MKS AWK comes with full technical and tutorial documentation to speed your mastering of this fourth generation programming language.

MKS AWK sells for **$75.**

Order both the software and the book from MKS for **$89.**

Also available:

The **MKS Toolkit**: over 110 UNIX-based tools for DOS including the **Korn Shell**, **Vi**, and **AWK**, complete with nearly 400 pages of documentation and tutorials.

The complete package: **$139.**

MKS **Vi**: the UNIX screen editor running under DOS at lightning fast speeds — it's tuned for the PC.

Comes with Tutorial and Reference Manual for **$75.**

Mortice Kern Systems Inc.,

43 Bridgeport Road East, Waterloo, Ontario, Canada, N2J 2J4 **(519) 884-2251**

uucp: {allegra, decvax, ihnp4}!watmath!mks!toolkit

MKS AWK runs under MS-DOS 2.0 or later. Not copy protected. Prices quoted in US funds. VISA, MASTERCARD, American Express, uucp, and purchase orders are accepted. Overseas orders please add $10 for postage and handling. MKS is a registered trademark of Mortice Kern Systems Inc. UNIX is a trademark of AT&T Bell Labs. MS-DOS is a trademark of Microsoft Corp.

CIRCLE 249 ON READER SERVICE CARD

(continued from page 51)

board. (Ray Duncan reports that Cruise Control also works.) FOKKER runs fine, albeit a little slowly, on a vanilla PC without QUICKEYS, but with anything faster than a 4.77-MHz 8088, it requires some sort of keyboard speedup program. The normal keyboard routines just don't read the stick routines fast enough, and the Fokker flies off the screen despite all efforts to catch him.

UR/FORTH users may want to use Tom Almy's Native Code Compiler to make the game faster on a standard PC, but it really doesn't need the extra speed on faster machines. On both Compaq 386 and Zenith 386 models, it is really too fast and probably ought to have a slow-down loop in the gun firing routines added. The rate of fire varies with the processor, also. An 80386 will fire at about 480 rounds per minute (per gun), a possible but high rate. (World War I aircraft synchronizers varied the rate of fire with engine speed.)

The Game Listing

The source is heavily commented. Probably the easiest way to understand the game is to examine the main loop definition on Screen 15. After setting initial values, display mode, and so on, *DOGFIGHT* enters a *BEGIN...AGAIN* loop. *DR.1* puts the Fokker bit-map address on the stack. Next, the current screen coordinates of the upper-left *CORNER* of the bit map are fetched to the stack. *MOVEFOKKER* leaves two random values within the current *RANGE* of random motion on the stack. These are vector-added to *CORNER*'s x and y. *?STICK* then determines if a cursor keypress is in the buffer and, if it is, leaves values appropriate to the key pressed on the stack; otherwise, it leaves a pair of 0s. The *?STICK* values are then vector-added to the previously "moved" corner values, and the Fokker is displayed at the new location by *!BLOCK*. What this does is randomly shift the Fokker's position and allow your keypress to counteract (somewhat) the random shift.

Next, *SHOOTING?* (defined in Screen 10) polls the keyboard and, if a key has been pressed, checks to see if it was the space bar. If it was, then if there is any *AMMO* left, it fires the guns, decrementing the ammo by two rounds, and calls *HIT?* (defined in Screen 9) to see if *CORNER* is within the x,y range that describes a hit in the Fokker's cockpit zone. If there is a hit, it's actually two hits (two guns, right?), and so *#HITS* is incremented by 2. With each four hits *EXCITEMENT* increments the *RANGE* of random motion and the Fokker pilot shouts a curse, in German, to distract you. To avoid offending anyone, I have made the curses in Screen 8 merely illustrative, but you can probably come up with somewhat more irritating and appropriate phrases.

Back in the *DOGFIGHT* loop, you now check *#HITS* again to see if the Fokker has been hit 20 times. If it has, it *EXPLODE*s with three *BURST*s, and you *WIN* and *EXIT*. If it hasn't been hit 20 times, you check the *AMMO* remaining. If it's zero, you *LOSE* and *EXIT*. Finally, you redisplay the *GUNSIGHT*.

The code for *?stick* in Screen 7 leaves –2s, 0s, or +2s depending on the key pressed. Initially, the *RANGE* of Fokker motion is set at –1 to +1. This means that you can outfly the

If You Have Turbo C You Have Half Your C-Programming Vehicle

Turbo C is a great compiler but there is one vital cog missing—debugging. Without it, you have to spend an awful lot of energy to go a short distance.

Gimpel Software's C-terp, long recognized as the leading C interpreter, now fully supports Turbo C with complete compatibility guaranteed.

Interactive Debugger—Our debugging facilities include split screen (code in upper portion, dialog in lower), breakpoints (sticky, temporary, line/function, cursor-directed), display of structures and arrays, execution of any expression (even those involving macros), function traceback with arguments, watch expressions and watch conditions (watchpoints). Our watch expressions can be structs or arrays. We catch out-of-bounds pointers!

No Toy—Full K&R with ANSI enhancements. Multiple-module with a built-in automatic make. It has virtual memory option (with optional direct use of extended memory) and a shared symbol option for those big programs. It supports graphics, dual displays and the EGA 43-line mode.

Links to external libraries—(both code and data, automatically) which can call back to interpreted functions. Function pointers are compiler compatible.

100% Turbo-C compatible.—Same header (.h) files, data alignment, bit field orderings and preprocessor variables as your compiler. We link in your compiler's library.

Our reconfigurable editor—is multifile and comes with a configuration script to mimic Turbo's editor.

The missing wheel that will turn your half-cycle into a bicycle

C-terp

Order C-terp today!
Call (215) 584-4261

Introductory Price for Turbo C-terp:
$139.00
VISA, MC, COD—30 day money back guarantee

C-terp Version 3.0 is also available for the following compilers:
Microsoft, Lattice, Aztec, C86, and Mark Williams ($298) and Xenix ($498).

GIMPEL SOFTWARE
3207 Hogarth Lane
Collegeville, PA 19426

C-terp is a trademark of Gimpel Software, and Turbo C of Borland International.

Fokker easily, at first. With each four hits in the fuselage of the triplane, however, *RANGE* expands by –1 and +1: after the first 4 hits, *RANGE* is –2 to +2; after 16 hits, it is –4 to +4—twice the range of control you have with the numeric keypad. Because you are making a conscious effort to move in one direction, and the Fokker is moving randomly, you can still get him in your sights after 16 hits (statistically). But he doesn't stay there very long!

Screen 1 loads UR/FORTH's DOS level-2 interface and builds the file-handling words. Screen 2 assumes you have a file containing the Fokker image, called DERFOKKR.IMG, in the same directory and loads it into memory. Note that *DR.1* allots 3,000 bytes whereas the file is only 2,866 + 16 (palette) bytes long.

There is something funny going on in LMI's sizing of arrays for *@BLOCK* and *!BLOCK*, and the "correct" values seem to pick up garbage somehow; the "oversized" 3,000 bytes is empirically adequate.

Availability

All the source code for articles in this issue is available on a single disk. To order, send $14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call (415) 366-3600, ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

For non-Forth programmers with an EGA display who want to pit their skill against the red Fokker, a complete unprotected run-time version (load-and-go .EXE file) is available from me for $14—cash, check, or M.O. only; no purchase orders, please. Have fun!

Bibliography

Campbell, Christopher. *Aces and Aircraft of World War I*. New York: Greenwich House, distributed by Crown Publishers Inc., 1984:139.

Cockerham, John T. "The EGA Standard." *PC Tech Journal*. vol. 4, no. 10 (October 1986): 49–79.

Hoffmann, Thomas V. "Graphic Enhancement." *PC Tech Journal*. vol. 3, no. 4 (April 1985): 58–77.

Mansfield, Victor. "Scientific Graphics with the EGA." *PC Tech Journal*. vol. 3, no. 9 (September 1985): 163–170. *IBM Personal Computer Seminar Proceedings*. vol. 2, no. 11 (November 1984).

Petzold, Charles. "Exploring the EGA, Part 1." *PC Magazine*. vol. 5, no. 14 (August 1986): 367–384.

Petzold, Charles. "Exploring the EGA, Part 2." *PC Magazine*. vol. 5, no. 15 (September 16, 1986): 367–384.

Ross, Hugh N. "Programming the Enhanced Graphics Adapter." *Exchange*. (September/October 1986): 14–17. International Business Machines Corp.

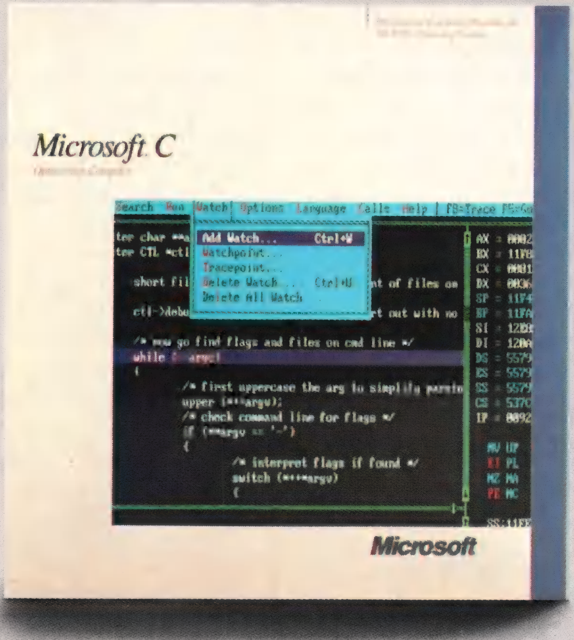
Wilton, Richard. "Programming the Enhanced Graphics Adapter." *Byte* vol. 10, no. 11 (Fall 1985).

DDJ

(Listings begin on page 88.)

Vote for your favorite feature/article. Cicle Reader Service **No. 4.**

C 5.0 has three features professional programmers can't live without.



Automated Interrupt Handling in C

by Ron Miller

Make your own TSR utilities

A good case could be made that anything that makes the writing of resident programs easier is socially counterproductive. There are too many of those darned things already. But at times it can be useful to hack out a quick resident utility to serve a transient purpose such as reconfiguring hardware on the fly or offering an extra help screen, in much the same way as it is sometimes useful to hack out a file filter or a printer configuration utility. If you need to do such things, doing them in a high-level language adds immeasurably to the ease and accuracy of the results.

Any MS-DOS programmer who writes memory-resident utilities or works with serial communications knows that coding interrupt handlers is a task that quickly becomes tedious without ever becoming routine. Debugging one more run of assembly language—especially some assembly language triggered by another program—inevitably burdens any project with considerable overhead. If only there were some way to get the requisite *PUSH*s, *POP*s, *CLI*s, *STI*s, and *MOV*s straight once and for all.

If a set of assembly-language routines for interrupt handling could be regularized and debugged and compiled, the modules could be stored away in a library for linking with high-level code used for the bulk of the interrupt handlers themselves.

Unfortunately, interrupt handlers are exceedingly specialized objects. Different registers must be set and preserved; different interrupts called; and in some cases, flags must be returned intact. The programmer who writes everything from scratch in assembly language at least retains absolute control over all registers and flags, but once you move into a high-level language such as C, the registers and flags cease to be easy to supervise. If every line of C must be scrutinized to determine what registers will be altered and what flags will be set, why not just stick with low-level coding?

I have, I believe, devised something close to a minimal system for automating the process so that interrupt handling, no matter how complex, can be taken out of the realm of assembly language and be made routine. Inevitably, there are patches of assembly language in the system, and some of it is compiler and/or memory model dependent. These routines, however, can be done once, then put away for linking on demand.

A Sketch of the Process

The key to successfully implementing this system is a two-step procedure in which a standardized low-level handler is invoked by the interrupt itself. This invariant first stage transforms the registers (and the stack, if necessary) so that the routine can act as a simple C function that calls another C function to do the actual work of the handler. This reestablishment of the C environment allows the high-level handler to use the full range of C syntax, to access the run-time package, and to employ automatic and previously initialized static variables. Variations from handler to handler can thus be confined to the far more easily maintained context of a high-level language.

Four linkable assembly-language routines are involved in every application:

1. A trivial initializing routine that stores the data segment of the C code in the code segment of the low-level handler.
2. A generalized low-level handler routine that:
   - *PUSH*es the registers
   - resets the *ds* register (and perhaps other registers) to the values needed for operation by the actual resident C code
   - calls the high-level handler itself
   - *POP*s the registers on return from the high-level handler
   - returns from the interrupt itself
3. An interrupt function that can address the operating system from within the high-level handler. As you will see, this function must use the register stack generated by the low-level handler routine as its data-in and data-out structure.
4. A routine that swaps the 32-bit address of the low-level interrupt handler for the interrupt vector being captured while moving the original vector to an unoccupied location in the interrupt table for calling or chaining.

Ron Miller, 1157 Ellison Dr., Pensacola, FL 32503. Ron is a regular contributor to Micro Cornucopia.

In the code fragments to follow, I neglect to set up the necessary *PROC*s, *ASSUME*s, and *SEGMENT*s for linking because those housekeeping details will vary from compiler to compiler. The *[bp+xx]* addressing may be displaced by a word or two from what you need, but a glance at a bit of assembly-language output from your compiler should reveal the necessary adjustments.

Storing the Data Segment Address

At the very least, for a handler's resident C code to work properly, the *ds* register must be reset to its original value. This is made possible by inserting a *saveds()* call somewhere in the initializing code before any vector swapping is carried out.

Coding interrupt handlers is a task that quickly becomes tedious without ever becoming routine.

The object library of the programmer should therefore contain a compiled version of the following code:

```
  PUBLIC saveds__,c__ds
c__ds dw 0 ;In-code storage slot for
                                DS.
saveds__: ;Or however your compiler/
                   assembler alters
        ;public names for assembly
                -language reference.
  mov cs:c__ds,
  ret
```

This assumes that your handler is not so complex that it requires a separate stack; if it does, *ss* and *sp* for the internal stack could also be squirreled away for recall. My advice, however, is to keep auto variables to the minimum needed for communication between C functions. Play storage games with static variables if considerable storage is needed. Keep it simple; use the other fellow's stack.

The Low-Level Interrupt Handler

The outline given earlier provides the rationale for the code in Listing One, page 100. This routine, labeled *Lhand1()*, provides the segment/offset address actually inserted into the interrupt table. In all the code to follow, I use the prefix *L* to tag low-level interrupt handlers and *H* to tag high-level ones.

Because some interrupts return information in the flags, you cannot use *iret* to end the routine. *FAR ret 2* strips the old flag off the stack to preserve the return. If this routine is not assembled as a *FAR* procedure, the *ret 2* must be replaced by *db 0cah,2,0* so that a *FAR* return to the caller is made. I urge you, if at all possible, to write your high-level handlers in a "large" C so that all calls and returns are *FAR* and so that the entire memory of the computer is available to operations using pointers. The small decrease in code size using a small-model C is more than paid for by the need to allocate peek and poke room in the data segment when *FAR* manipulations must be made.

Speed.

Fast Execution Speed.

	Microsoft® C 4.0	Microsoft C 5.0
Sieve (25 iterations)	5.7	3.3
Loop	11.0	0.0*
Float	19.9	0.1
Dhrystone	22.8	19.1
Pointer	14.2	7.4

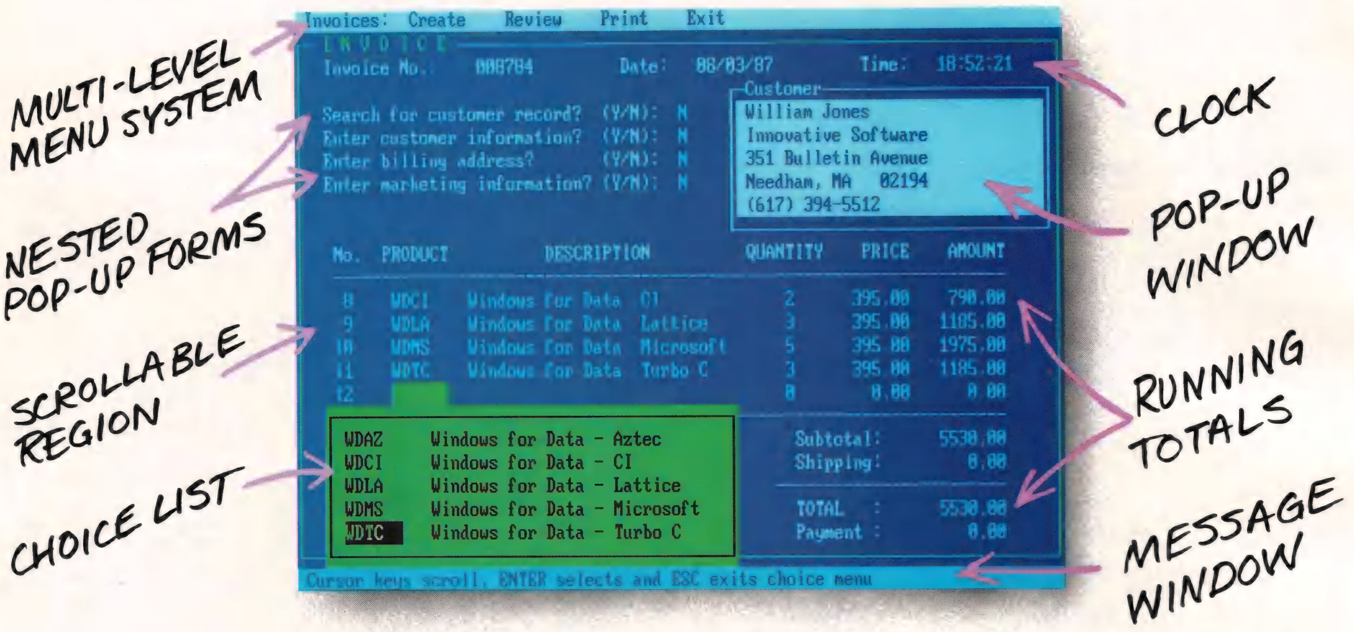
- New optimizations generate the fastest code:
  - —Inline code generation. **NEW!**
  - —Loop optimizations: **NEW!**
    - —Loop invariant expression removal. **NEW!**
    - —Automatic register allocation of variables. **NEW!**
  - —Elimination of common sub expressions.
  - —Improved constant folding and value propagation.
- Fine tune your programs for even greater speed:
  - —Coding techniques for writing the fastest possible programs are included in the documentation. **NEW!**
  - —Segment Allocation Control:
    - —Group functions into the same segment to get faster NEAR calls. **NEW!**
    - —Specify which segments receive variables to yield faster NEAR references. **NEW!**
  - —Uses register variable declarations.
  - —Mix memory models using NEAR, FAR & HUGE pointers.

*Time is negligible.

Microsoft C 5.0
Optimizing Compiler

WINDOWS FOR DATA®

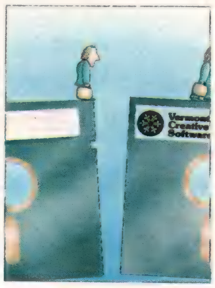
Uncommon Screens



```
Invoices:  Create    Review    Print    Exit
 I N V O I C E
 Invoice No.    000784        Date:   08/03/87        Time:   10:52:21
                                             Customer
 Search for customer record?  (Y/N):  N      William Jones
 Enter customer information?  (Y/N):  N      Innovative Software
 Enter billing address?       (Y/N):  N      351 Bulletin Avenue
 Enter marketing information? (Y/N):  N      Needham, MA   02194
                                             (617) 394-5512

  No.  PRODUCT          DESCRIPTION           QUANTITY    PRICE    AMOUNT

   8   WDCI   Windows for Data  CI               2       395.00    790.00
   9   WDLA   Windows for Data  Lattice          3       395.00   1185.00
  10   WDMS   Windows for Data  Microsoft        5       395.00   1975.00
  11   WDTC   Windows for Data  Turbo C          3       395.00   1185.00
  12                                             0         0.00      0.00

   WDAZ    Windows for Data - Aztec             Subtotal:         5530.00
   WDCI    Windows for Data - CI                Shipping:            0.00
   WDLA    Windows for Data - Lattice
   WDMS    Windows for Data - Microsoft         TOTAL    :        5530.00
   WDTC    Windows for Data - Turbo C           Payment  :           0.00
Cursor keys scroll, ENTER selects and ESC exits choice menu
```

If you program in C, take a few moments to learn how Windows for Data can help you build a state-of-the-art user interface.

- ☐ Create and manage menus, data-entry forms, context-sensitive help, and text displays — all within windows.
- ☐ Provide a common user interface for programs that must run on different machines and operating systems.
- ☐ Build a better front end for any DBMS that has a C-language interface (most popular ones do).



FROM END TO BEGINNING

Windows for Data begins where other screen packages end, with special features like nested pop-up forms and menus, field entry from lists of choices, scrollable regions for the entry of variable numbers of line items, and an exclusive built-in debugging system.

NO WALLS

If you've been frustrated by the limitations of other screen utilities, don't be discouraged. You won't run into walls with Windows for Data. Our customers repeatedly tell us how they've used our system in ways we never imagined — but which we anticipated by designing Windows for Data for unprecedented adapatability. You will be amazed at what you can do with Windows for Data.

YOU ARE ALWAYS IN CHARGE

Control functions that you write and attach to fields and/or keys can read, compare, validate, and change the data values in all fields of the form. Upon entry or exit from any field, control functions can call up subsidiary forms and menus, change the active field, exit or abort the form, perform almost any task you can imagine.

OUR WINDOWS WILL OPEN DOORS

Our windows will open doors to new markets for your software. High-performance, source-code-compatible versions of Windows for Data are available for **PCDOS** (**OS/2** soon), **XENIX**, **UNIX**, and **VMS**. PCDOS versions are fully compatible with Microsoft Windows, TopView, and DESQview. **No royalties**.

You owe it to yourself and your programs to try Windows for Data. If not satisfied, return for a full refund. Call for **FREE DEMO**.

Vermont Creative Software

21 Elm Ave., Richford, VT 05476
Telex: 510-601-4160 VCSOFT   FAX 802-848-3502
Tel.: 802-848-7731   ext. 31
Prices: PCDOS* $395; XENIX, VMS, UNIX Call.

CIRCLE 157 ON READER SERVICE CARD

(continued from page 55)

You'll note that the routine calls an *EXTERN Hhand1__*. Naturally, you must name your high-level handler *Hhand1( )* so that the linker can find it. The *1* in the name allows you to place several versions—*Lhand1( )*, *Lhand2( )*, *Lhand3( )*, and so on—into your library for use in complex programs involving several interrupt handlers. As long as each low-level handler calls its proper high-level partner, there isn't any confusion.

As you will see in the next section, the power of this strategy depends upon having the call to the high-level handler immediately preceded by the push-the-registers statements. If you insist upon setting up your own stack, swap *ss* and *sp* before the *PUSHf* and after the *POPf* so the register stack also serves as the stack for the C routine.

The High-Level Handler

The design of the high-level handler depends on an interesting feature of the C language: setting up the stack before, and cleaning up the stack after a function call, are the responsibilities of the calling routine, not of the function itself. C does this, I gather, to allow for the passing of a variable number of arguments. One secondary consequence of this design is that your higher-level function can be fooled into treating the stack it inherits as if it contained function arguments. Thus, if you give your C code handler a fake argument such as:

```
Hhand1(fake)
   int fake;
{. . .}
```

the compiler will treat the 2-byte region containing the pushed value of the *ax* register as though it were an integer that had been passed to the function. In effect, the entire stack of pushed register values becomes available to the high-level handler as an automatic variable.

If within the handler you declare:

```
typedef struct {
   int ax,bx,cx,dx,di,si,bp,ds,es,flags:
                                   int;
} REGS;
REGS *regs;
```

and then initialize *regs* to point to the base of the stack:

```
regs = &fake;
```

the stack is mapped to that structure. (In practice, of course, this structural definition would be handled globally with an *#include* statement.)

Want to set *dx* on the stack to 0? Write *regs->dx = 0;* without leaving C. After the higher-level routine is exited and the low-level routine *POP*s the stack it had *PUSH*ed, the original calling routine will see a return of 00 in *dx*. It doesn't matter what the C routine has done to the actual value of *dx* in the meantime, because upon *POP*ping the registers, the low-level routine will restore the old values—unless some purposeful changes have been made to the virtual structure **regs*.

The value of this ploy becomes clearer with a specific example. Suppose you wish to write a resident program that captures interrupt *16h*

Speed.

Fast Compilation.
Fast Prototyping.

Microsoft C Version 5.0 includes QuickC,™ which lets you edit, compile, debug, and execute in an integrated environment. It's ideal for prototyping.

- In-memory compilation at 10,000 lines/ minute. **NEW!**
- Built-in editor with parentheses, bracket and brace matching.
- Use the integrated debugger to animate through your program, add watch variables and set dynamic breakpoints. **NEW!**
- MAKE file is automatically generated for you. Simply indicate the modules you want to use, then MAKE recompiles and links only those modules that have changed. **NEW!**
- Full C 5.0 compatibility:
  - —Completely source and object code compatible.
  - —Emits CodeView®-supported executables.
  - —Identical compile/link command line switches.

Microsoft C 5.0
Optimizing Compiler

EVEN MORE POWER AND FLEXIBILITY

BRIEF 2.0

Users and industry press alike have unanimously proclaimed BRIEF as the best program editor available today. Now, the best gets better, with the release of BRIEF 2.0.

Straight from the box, BRIEF offers an exceptional range of features. Many users find that BRIEF is the only editor they'll ever need, with features like real, multi-level Undo, flexible windowing and unlimited file size. But BRIEF has tremendous hidden power in its exclusive macro language. With it, you can turn BRIEF into your own custom editor containing the commands and features you desire. It's fast and easy.

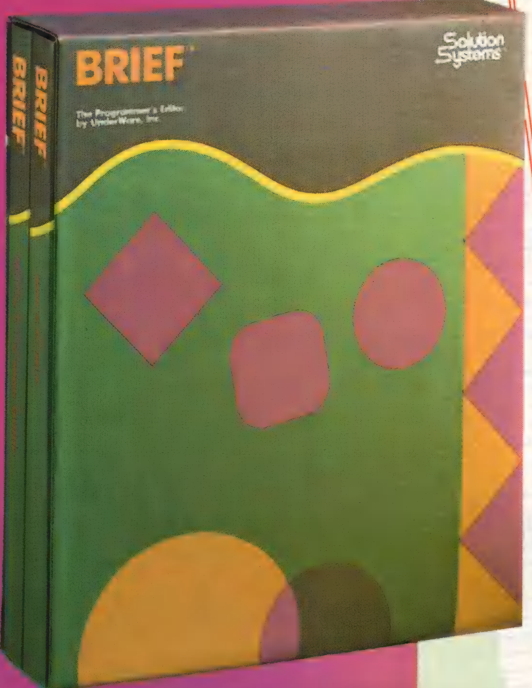
Jerry Pournelle, columnist for BYTE magazine summed it all up by saying BRIEF is, "**Recommended. If you need a general purpose PC programming editor, look no further.**" His point of view has been affirmed by rave reviews in C JOURNAL, COMPUTER LANGUAGE, DR. DOBB'S JOURNAL, DATA BASED ADVISOR, INFOWORLD AND PC MAGAZINE.

One user stated "BRIEF is one of the few pieces of software that I would dare call a masterpiece." Order BRIEF now and find out why. **BRIEF 2.0 is just $195. If you already own BRIEF, call for upgrade information.**

TO ORDER CALL: 1-800-821-2492 (in MA call 617-337-6963)

As always, BRIEF comes with a 30 day money-back satisfaction guarantee.

Solution Systems™
541 Main Street
Suite 410D
So. Weymouth, MA 02190
(617) 337-6963



Look at these BRIEF 2.0 enhancements!

Main Features:

- All new documentation with tutorials on basic editing, regular expressions **and** the BRIEF Macro Language.
- Setup program for easy installation and configuration. (Requires no knowledge of the macro language)
- Increased speed for sophisticated operations like Undo and Regular Expression Search.
- Expanded regular expressions, with matching over line boundaries.
- More block types, with marking by character, line or column.
- Command line editing (move cursor, add and delete characters, specify command parameters).
- Support for more programming languages.
- Optional borderless windows.
- Enhanced large display support, including wider displays.
- Reconfigurable indenting for C files (supports most indenting styles).

Plus the basic features that made BRIEF **SO** popular!

Basic Features:

- Full multi-level Undo
- Windows
- Edit many files at once
- File size limited only by disk space
- Automatic language sensitive indentation

Requires an IBM PC or compatible with at least 192K RAM.
BRIEF is a trademark of UnderWare, Inc.
Solution Systems is a trademark of Solution Systems.

CIRCLE 142 ON READER SERVICE CARD

(continued from page 57)

and checks for five different hot keys that trigger five alternative routines. You would put a long pointer to *Lhand1()* in the place of *int 16h* in the interrupt table and move the old *int 16h* vector out at *NEW16*. The higher-level routine could be written as in Listing Two, page 100.

This, I submit, is considerably easier to write and maintain than an equivalent assembly-language routine full of *CMPs*, labels, and leapfroggings.

The Interrupt Function

With luck, your particular version of C offers an "interrupt" function that permits the stack manipulations described in the previous section. The requirements are:

- the use of a pointer-to-a-register-structure as an argument for the calling function
- the inclusion of all data-bearing registers (including a separate "flag" integer) in that structure
- the potential for using a single memory region as both **inregs* and **outregs*, to use the old Lattice *int86()* terminology. In this case all you need to do is be sure to *PUSH* and *POP* the registers in the low-level handler in exactly the order ordained by the function definition you have inherited.

If, as seems likely, your compiler has made other choices, you can find a suitable version for assembling in Listing Three, page 100. There is probably no need to analyze the code at length, except to observe that the version offered assumes 32-bit pointers and a need to preserve *ds* and *bp* across the function call. Anyone who will profit from automating interrupt handling will be able to check his or her own compiler's assembler output and modify the base pointer addressing to fit the brand and the memory model at hand. The basic strategy is to load the registers from the structure pointed at, make the call in question, reload the structure with the returning values in the registers, and exit.

It is worth noting that the interrupt function in Listing Three has the added virtue of being reentrant —which can be significant if you're capturing more than one interrupt vector in a resident program. Earlier drafts of this system broke down because I tried *movsb*ing parts of the stack back and forth to the global array used by my compiler's interrupt function. Why not adapt the situation, I reasoned, to the ready-made function? Things went swimmingly in simple hot-key routines. When I used the clock interrupt to poll the DOS-is-interruptable flag given by interrupt *21h* service *34h*, however, the register stacks for various interrupts began to corrupt one another 18.2 times a second. Moreover, the compiler's routine did not return the raw flags but provided separate carry and zero flag Booleans. I found myself having to *AND* and *OR* the stack to set up the return. It seemed simpler to write my own function.

The Interrupt Vector Swapping Routine

The code for the interrupt vector

And speed.

Fast Debugging.

Microsoft C Version 5.0 includes Microsoft CodeView, our source-level windowing debugger that lets you debug more quickly and thoroughly than ever before.
- Debug larger programs:
  - —Debug through overlays created by the Microsoft overlay linker. **NEW!**
  - —Expanded Memory Specification (EMS) support. **NEW!**
- Fast debugging through precise control of your program execution:
  - —Access source level and symbolic debug information from your Microsoft C, FORTRAN, and Macro Assembler programs. **NEW!**
  - —View your source code and assembly simultaneously.
  - —Watch the value of variables change as you execute.
  - —Set conditional breakpoints.
  - —Animate or single step through your program.
- CodeView brings you as close as you've ever been to your hardware:
  - —Swap between your code and output screens.
  - —Watch your registers and flags change as your program executes.

All benchmarks run on an IBM® Personal System/2.™

NOW SHIPPING

Microsoft®

For your free C 5.0 information packet, call:
(800) 426-9400.

In Washington State and Alaska, (206) 882-8088. In Canada (416) 673-7638. Microsoft, the Microsoft logo and CodeView are registered trademarks and QuickC is a trademark of Microsoft Corporation. IBM is a registered trademark and Personal System/2 is a trademark of International Business Machines Corporation.

CIRCLE 380 ON READER SERVICE CARD

Instant Replay

Version III

See us at
COMDEX/Fall '87
November 2–6, 1987
Bally's Las Vegas B210
Las Vegas, Nevada

SOFTWARE AHEAD OF ITS TIME

"Instant Replay™ is one of those products with the potential to go from unknown to indispensable in your software library." *PC Magazine*

"Incredible . . . We built our entire Comdex Presentation with Instant Replay.™" *Panasonic*

"Instant Replay™ brings new flexibility to prototypes, tutorials, & their eventual implementation." *Electronic Design*

"I highly recommend Instant Replay.™" *Computer Language*

"You need Instant Replay™!" *Washington Post*

***Instant Replay™* is a unique authoring system for creating:**

Demonstrations · Tutorials · Music Presentations · Prototypes · Menus

When active, *Instant Replay™* is a DOS Shell that runs, memorizes, and replays programs. It has the unusual ability to insert prompts, pop-up windows, prototypes, user involvement, music, and branching menus into replays.

Building a Demo or Tutorial is easy. Just run any program with *Instant Replay™* and insert explanation pop-up windows, prompts, user involvement, music and so on. *Instant Replay™* remembers everything; it builds a demo that will re-run the actual program or a screens only prototype version.

Instant Replay™ includes a *Screen Maker* for building pop-up windows, prototype windows, and menu windows. Other useful tools include a *Prototyper, Keystroke Editor, Music Maker, Menu Maker, Presentation Text Editor, Control Guide, and Insertion Guide.*

The screen editor *Screen Genie™* is designed to be memorized by *Instant Replay.™* Creating animations is easy and fun; just run *Screen Genie™* and memorize your activity.

Instant Replay™ for IBM and True Compatibles, requires DOS 2.0 or greater. *Instant Replay™* is not copy protected. **There are no royalties required for distribution of Demos.**

Instant Replay™ at $149.95 is an exciting new product. Because of the quality of this product, *Nostradamus©* provides a 60-day satisfaction money back guarantee. Call or write, we accept VISA, AmEx, C.O.D., Check or P.O. with orders. Demo diskettes and free product brochure available.

Nostradamus, Inc. 3191 S. Valley Street, (ste. 252) Salt Lake City, Utah 84109
voice (801) 487-9662
Data/BBS 801-487-9715 1200/2400,n,8,1

Instant Replay™ Features:

- *Illustrated manual with index*
- *Music Maker*
- *Text Editor*
- *Online Help*
- *Screen Painter*
- *Magic Demo Animator*
- *Dynamic Menu Maker*
- *Memorize and Replay actual programs*
- *Memorize and Replay screens only*
- *Insert: prompts, pop-ups, prototypes, music, and user involvement into replays*
- *Make insertions while creating or reviewing*
- *Generate Vapor Ware from actual programs*
- *Resident Screen Painter for creating and grabbing windows on the fly*
- *Prototyper that includes slide shows, menus, and nesting*
- *Keystroke/time editor, inserter, and merger*
- *Replay chaining and linking*
- *Modular demo making facilities*
- *Fast forward and single step modes*
- *Self Made Tutorial included*
- *Timed Keyboard Macros*
- *Numerous and powerful operator input options for Tutorials*
- *Text File Presentation facility*
- *Transparent Windows*
- *Change Defaults*
- *Foreground or Background Music*
- *Canned special sound effects*
- *Unlimited replay branching*
- *Compressed screens*
- *Object oriented programming*
- *Tracking editor*
- *Plus much more . . .*

Instant Replay™
Instant Assistant™
Screen Genie™
Word Genie™
NoBlink Accelerator™
Assembler Genie™
DOS Assistant™
Programming Libraries
Supports Turbo Pascal 4.0, Turbo C, MS Quick C
HardRunner™
. . . and more

Nostradamus®

CIRCLE 251 ON READER SERVICE CARD

INTERRUPT HANDLING
(continued on page 59)

swapping routine is in Listing Four, page 101. It uses *int 21h*, functions *35h* and *25h*, to get and put interrupt vectors. By setting up a standard function that uses the old interrupt number, the new (or chained) interrupt number, and a pointer to the low-level handler, you can carry out the entire process of vector swapping without explicit recourse to assembly language. Notice that the swap routine returns nonzero in *ax* (modify it if *ax* is not your integer-return register) if the chaining interrupt is in use. Such a feature could be useful in a find-an-unused-interrupt routine.

Once again I assume 32-bit pointers and preserve *bp* and *ds*. A small-model C routine would require more explicit loading of segment registers into *ds* before using service *25h*. In any case, the name of the low-level interrupt handler (in this case, *Lhand1*) is used as the final function argument because C compilers treat a function name by itself as a function pointer—32- or 16-bit—depending on the memory model.

Setting Things Up

If the four functions I've discussed have been compiled and stored in an object file library, setting up a resident program is scarcely more complicated than the writing of the handler code itself. Listing Five, page 102, provides a template for initializing a generic resident program. For clarity's sake I capture a single interrupt; however, there is no limit to the number of vectors that can be inserted into the table with multiple *chgint( )*s. Certainly, in production software the error-handling would be more complex.

To estimate the program size, I ordinarily use interrupt *21h*, service *51h*, to obtain the PSP address of the C program, then subtract that number from the paragraph address of the top of the static heap —plus a couple more paragraphs just to be on the safe side.

DDJ

(Listings begin on page 100.)

Vote for your favorite feature/article
Circle Reader Service **No. 5.**

Dr. Dobb's Journal

Subscription Problems?

No Problem!

Give us a call and we'll straighten it out. Today.

Outside California
CALL TOLL FREE: 800-321-3333
Inside California
CALL: 619-485-6535 or 6536

And here's the speediest way to get C5.0:

Just visit any Egghead store. There's more than 70 nationwide. And each one has C5.0 in stock, at a discount price.

Or, just give us a call on our toll-free number. We'll send you C5.0 by mail. Within 72 hours.

It's the fastest way we know to get a great deal on a very fast compiler.

Call 1-800-EGGHEAD

EGGHEAD Discount SOFTWARE℠

An Alternative to Soundex

Upping the word-matching hit rate

by Jim Howell

. . . if things had to be identical before you could recognize them, you would never recognize anything at all. . . . The fact is that our ability to work and to act in the real world depends on our accepting a "tolerance" in our recognition and in our language.
—Jacob Bronowski, A Sense of the Future

Much of the art of communication relies on the ability to differentiate what is meant from what is said. This goes far toward explaining why I have so much trouble at the keyboard; computers have none of the tolerance that a bad (and slightly dyslexic) typist such as myself needs. My computer doesn't understand that "dri" means "dir," so it yells "Bad command or file name," then sits there, waiting for something intelligible.

To you and me, it's obvious what command I actually meant, but to the computer, lacking the recognition tolerance shaped by millions of years of evolution, the input is gibberish. As simple as this example seems, it has proved a difficult problem to solve through some mechanism that a computer can use efficiently. In spite of the time and effort that have gone into attempts to solve this problem (the venerable Soundex algorithm dates from World War I), no wholly satisfactory solution has yet come to light.

But that isn't to say that there haven't been gains on the problem. In fact, another promising solution has recently arisen that is certainly worth passing on, warts and all. The algorithm comes from Michael Bickel and was published in the March issue of the *CACM*.[1]

The method is simple—each letter has a weight associated with it. The "likeness score" between two words is calculated by summing the weights of all the letters in common between the two words, each letter being counted only once. Letters are converted to uppercase, and numbers, punctuation, and white space are ignored. The method can be easily made sensitive to case or to consider numbers.

The weights are based on the frequencies of the letters, being the negative of the logarithm (base 2) of the frequency of the letter (the frequencies sum to 1.0). This gives greater weight to the less common letters. Bickel's weights are given in Table 1, below.

Weight	Letters
3	a e i n o s t
4	d h l r u
5	c f g m p w
6	b v
7	k q
8	j, x, y
9	z
0	all others

***Table 1:** Letter weights used in Bickel's algorithm*

Bit	7	6	5	4	3	2	10
Byte 0	–	a	e	i	n	o	st
Byte 1	–	–	–	d	h	l	ru
Byte 2	–	–	c	f	g	m	pw
Byte 4	b	v	k	q	j	x	yz

***Table 2:** Positions in the letter set*

As an example, say you want to match the word *ecdysiast*. You first create a set (I call this a letter set; Bickel uses the name *MASK*)—*acdeisty*—from the letters in the word. Now consider the word *ecstasy*, which has the letter set *acesty*. To compare these, take the intersection of the two sets—namely, *acesty*. The first letter in the new set is *a*. The weight of this letter is 3, so the likeness score is set to this. The next letter is *c*, so you add 5 to the score. Then add 3 for *e*, 3 for *s*, 3 more for *t*, and 8 for *y* to give a likeness of 25.

To compare *ecdysiast* to *ectoplasm*, you form the letter set *acelmopst*. The intersection with the set of *ecdysiast* is *acest*, and so the likeness score of these two words is only 17. If you were to transpose two letters and compare *edcysiast*, say, the likeness scores would still be 25 and 17. The score of comparing *ecdysiast* and *edcysiast* would be 32, however, so *ecdysiast* becomes the word to look at more closely.

Implementation

Bickel illustrated the algorithm using NOMAD2 (a fourth-generation language), but it can just as easily be implemented in C using a bit set with 26 members. For each word, set the appropriate bit for each letter, leaving the rest unset. Then combine the two sets with a logical AND. Only the common letters are left in the combined set, which are then added using the appropriate weights.

You can do this in several ways. I chose to use a 4-byte array (of type *unsigned char*) to hold the letter set. A static data array, which is indexed by

Jim Howell, 45 Lodgepole Dr., Evergreen, CO 80439. Jim is a former geologist who now works as a programmer for Reference Technology in Evergreen.

lowercase letters, is initialized with the index to the appropriate byte in the letter set array along with the mask necessary to set the correct bit (see Listing One, page 110). The letter set array is designed so that the first three bytes have common weights, with the rest being stuffed into the fourth byte (see Table 2, page 54).

To construct the letter set for a word, go through the word character by character. If the character is a letter, convert it to lowercase and use this as an index into the data array to set the appropriate bit in the letter set array. Duplicate letters result in needless repetition; once a bit is set, setting it again doesn't do any good, but any scheme to avoid setting an already set bit would have too much overhead, so the problem, such as it is, is ignored.

Calculating the likeness score from the two letter sets is straightforward. The first three bytes use a common algorithm. Combine the appropriate bytes from the two letter sets into a mask with a logical AND; then examine the rightmost bit of the resulting byte, and if it is set, increase the likeness score by the appropriate weight; and finally, shift the mask to the right one bit and examine the rightmost byte, and so on through the bits, using a *for* loop.

The fourth byte is more complicated because its letters have different weights. The easiest solution is to operate on each bit individually, in effect hard-wiring the weights. Speed is most important here, space and elegance less so.

As an example, let's search the set of keywords in Table 3, right, using several common (for me) misspellings of *geochemistry.* Using *geochemsitry* (*s* and *i* transposed), the highest likeness scores are:

geochemistry	(46)
biochemistry	(41)
geochronometry	(40)
geothermics	(38)
geophysical	(34)

A search with another misspelling *geochemistru* (*u* mistyped for *y*)—gives:

geochemistry	(38)
geothermics	(38)
biochemistry	(33)
geochronometry	(32)
geochemical	(28)

And using *geochemistr* (*y* dropped), you have:

geochemistry	(38)
geothermics	(38)
biochemistry	(33)
geochronometry	(32)
geochemical	(28)

If the search word is simply *chemsitry* (with the transposition), the best matches are:

geochemistry	(38)
biochemistry	(38)
geochronometry	(32)
geothermics	(30)
geophysical	(26)

The last three examples point out the main weakness of this method: letter order is not significant. To us, *geochemistr* is nothing like *geothermics,* but the method is too simple to detect this. Other algorithms can check for transpositions, or altered letters, or dropped letters,[2] but because of combinatorial complexities, these can be slow. The strength of Bickel's algorithm lies in its simplicity, which in practice means speed. It can be used as a first pass to select likely candidates for the slower routines.

Limitations

Bickel's algorithm has some trouble with short words, especially if the letters are high-frequency ones. Such a word could match a great many other words—a major problem if a given word list is long. In fact, any word composed of frequent letters behaves in this way, no matter what the length. The longer the list to be searched, the greater the problem.

As an example, a search of the keyword list using *meter* gives:

geochemistry	(15)
geothermics	(15)
geochronometry	(15)
biochemistry	(15)
geothermal	(15)
geometry	(15)

In a long word list, the search should be restricted—for example, by assuming that the first letter is correct. Even this may not be enough, as with *George:*

geomorphology	(15)
geochemistry	(15)
geochronology	(15)
geochronometry	(15)
geopressure	(15)
geographic	(15)
geothermal	(15)
geothermics	(15)
geometry	(15)

Here additional criteria, such as having the lengths of the words be "close," could be profitably used to narrow down the search.

geochemical
geochemistry
geochronology
geochronometry
geodesy
geographic
geologic
geological
geomac
geometry
geomorphology
geophysical
geophysics
geopressure
geothermal
geothermics
biochemistry

Table 3: *Keywords taken from a reference list.*

Thoughts on Optimization

In a real application of this algorithm, you would probably want to build it for speed, which will mean a bout with the assembler. But preplanning can help. If there are only a handful of words in the list to be searched, any savings in time will likely be too small to be noticed by the user. Also, you could save some time by constructing the letter sets in advance so that speed wouldn't be as important. Fast code is most essential when there is a large list to search and in interacting with an impatient user.

For a fast assembler routine, I would unfold all the loops, making the code bulky but avoiding all the overhead of incrementing and comparing the counter. The masks and weights would be built in as immediate data, again sacrificing space for speed. Order is not important here, so by putting the least common letters at the end, it would be possible to check for an empty set and return

YOUR SYSTEM'S KEY COMPONENT

$3.95 U.S.A.
March/April 1987
Vol. 3 No. 2
Micro/Systems Journal
For the Advanced Computer User
Multitasking Between Programs
see page 22
Also in this issue
Communicating at 9600 bps . . 32
Program Interfacing to Microsoft Windows . . 36
Classic Technology's 286 Speed Pak . . 45
and introducing the column
LANScape . . 76

The Only Magazine By And For Advanced Micro Users.

At last there is a magazine that brings you the strictly technical but practical information you need to stay up-to-date with the ever changing microcomputer technology . . . *Micro/Systems Journal. Micro/Systems Journal* is written with the needs of the systems integrator in mind—the individual who's involved in putting together the hardware and software pieces of the microcomputer puzzle.

In each issue of *Micro/Systems Journal* you'll find such useful and progressive articles as:

- Interfacing to Microsoft Windows
- Unix on the PC
- 80386 Programming
- High Resolution PC Graphics
- Using 80286 Protected Mode
- Multiprocessing and Multitasking

You'll get the hands-on, nuts and bolts information, insight and techniques that *Micro/Systems Journal* is famous for . . . in-depth tutorials, reviews, hints . . . the latest information on computer integration, networks and multi-tasking, languages, and operating systems . . . hard-hitting reviews.

To start your subscription to *Micro/Systems Journal*, simply fill out one of the attached cards or write to *Micro/Systems Journal*, 501 Galveston Dr., Redwood City, CA 94063. You'll receive a full year (6 issues) of *Micro/Systems Journal* for just $20, and enjoy the convenience of having M/SJ delivered to your doorstep each month. Don't wait . . . subscribe today!

MICRO/ SYSTEMS JOURNAL

FOR THE ADVANCED COMPUTER USER

SUBSCRIBE NOW AND

SAVE OVER 47%

OFF THE NEWSSTAND PRICE!

47% SAVINGS

Yes! I want to subscribe to Micro/Systems Journal™ and save over 47% off the cover price.

☐ **1 Year (12 issues) $29.97** ☐ **2 Years (24 issues) $56.97**

Please charge my: ☐ Visa ☐ Master Card ☐ American Express

☐ Payment Enclosed ☐ Bill me later

Card # ______________________ Exp. date __________

Signature ______________________

Name ______________________

Address ______________________

City ______________________ State ________ Zip ________

Savings based on a full one-year cover price of $47.40. Canada and Mexico add $6 for surface mail, $14 for airmail per year. All countries add $24 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A PUBLICATION OF M&T PUBLISHING, INC. 2002

47% SAVINGS

Yes! I want to subscribe to Micro/Systems Journal™ and save over 47% off the cover price.

☐ **1 Year (12 issues) $29.97** ☐ **2 Years (24 issues) $56.97**

Please charge my: ☐ Visa ☐ Master Card ☐ American Express

☐ Payment Enclosed ☐ Bill me later

Card # ______________________ Exp. date __________

Signature ______________________

Name ______________________

Address ______________________

City ______________________ State ________ Zip ________

Savings based on a full one-year cover price of $47.40. Canada and Mexico add $6 for surface mail, $14 for airmail per year. All countries add $24 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A PUBLICATION OF M&T PUBLISHING, INC. 2002

47% SAVINGS

Yes! I want to subscribe to Micro/Systems Journal™ and save over 47% off the cover price.

☐ **1 Year (12 issues) $29.97** ☐ **2 Years (24 issues) $56.97**

Please charge my: ☐ Visa ☐ Master Card ☐ American Express

☐ Payment Enclosed ☐ Bill me later

Card # ______________________ Exp. date __________

Signature ______________________

Name ______________________

Address ______________________

City ______________________ State ________ Zip ________

Savings based on a full one-year cover price of $47.40. Canada and Mexico add $6 for surface mail, $14 for airmail per year. All countries add $24 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A PUBLICATION OF M&T PUBLISHING, INC. 2002

No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

Micro/Systems Journal

Box 3713
Escondido, CA 92025-9843

No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

Micro/Systems Journal

Box 3713
Escondidio, CA 92025-9843

No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

Micro/Systems Journal

Box 3713
Escondidio, CA 92025-9843

SOUNDEX ALTERNATIVE
(continued from page 63)

early from the function if there was nothing more to check.

It is possible to save some space and a little speed by eliminating the two most common letters, *e* and *t*. The letter set could then be stored in three bytes, and two nearly inevitable comparisons could be eliminated from the likeness calculation.

Comparison with the Soundex Algorithm

The Soundex algorithm is a creaking ancient by computer standards, but because it is still widely used, it is worth comparing it with Bickel's new algorithm. To review (or introduce) it, the Soundex algorithm constructs a code by preserving the first letter of a name. From the rest, eliminate all vowels and the letters *h* and *w* along with all doubled letters (keeping one of them). The remaining letters (except the first) are replaced with numbers. The key is the initial letter and the first three numbers of the rest of the name, padded with 0s if necessary.[3]

The main advantage the Soundex algorithm has over Bickel's is that the computation is done only once per word. The key is more like a hash function in that it is usually arbitrary and there is no good way to compare two values. Bickel's algorithm requires a set of calculations at each comparison but does a good job of indicating the amount of similarity.

Notes

1. Michael Allen Bickel, "Automatic Correction to Misspelled Names: A Fourth-Generation Language Approach," *CACM* 30 (March 1987): 224–228.
2. Dave Taylor, "Wordz That Almost Match," *Computer Language* 3 (November 1986): 47–59.
3. Donald E. Knuth, *The Art of Computer Programming, Volume 3: Searching and Sorting* (Reading, Mass.: Addison-Wesley, 1973): 391–392.

DDJ

(Listing begins on page 98.)

Vote for your favorite feature/article.
Circle Reader Service **No. 6.**

The Advanced Programmer's Editor That Doesn't Waste Your Time

For DOS, Microport UNIX, or . . . OS/2 Protected Mode

EPSILON

- Fast, EMACS-style commands—completely reconfigurable
- Run other programs without stopping Epsilon—concurrently!
- C Language support—fix errors while your compiler runs
- Powerful extension language
- Great on-line help system
- Multiple windows, files
- Regular Expression search
- Unlimited file size, line length
- Supports large displays
- 30 day money-back guarantee
- Not copy protected

Only $195

Lugaru Software Ltd.
5740 Darlington Road
Pittsburgh, PA 15217

Call (412) 421-5911
for IBM PC/XT/AT's or compatibles

CIRCLE 135 ON READER SERVICE CARD

FULL AT&T C++ for half the price of our competitors!

Guidelines announces its port of **version 1.1** of AT&T's C++ translator. As an object-oriented language, C++ includes: classes, inheritance, member functions, constructors and destructors, data hiding, and data abstraction. 'Object-oriented' means that C++ code is more readable, more reliable and more reusable. And that means faster development, easier maintenance, and the ability to handle more complex projects. C++ is **Bell Labs' answer to Ada and Modula 2.** C++ will more than pay for itself in saved development time on your next project.

C++

from GUIDELINES for the IBM PC: $195

Requires IBM PC/XT/AT or compatible with 640K and a hard disk.
Note: C++ is a *translator*, and requires the use of Microsoft C 3.0 or later.

Here is what you get for $195:
- The full AT&T v1.1 C++ translator.
- Libraries for stream I/O and complex math.
- "The C++ Programming Language", the definitive 327-page tutorial and description by Bjarne Stroustrup, designer of C++.
- Sample programs written in C++.
- Installation guide and documentation.
- 30 day money back guarantee.

To order:
send check or money order to:

**GUIDELINES SOFTWARE
P.O. Box 749
Orinda, CA 94563**

To order with Visa or MC, phone (415) 254-9393.
(CA residents add 6% tax.)

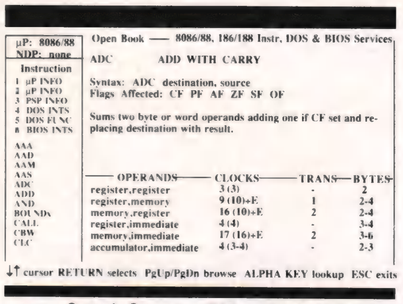
C++ is ported to the PC by Guidelines under license from AT&T.
Call or write for a free C++ information package.

CIRCLE 351 ON READER SERVICE CARD

The PCTALK family now introduces:

"A Better Mousetrap In Pop Up C and Assembly Information Reference Guides"

OPEN BOOK™



```
µP: 8086/88   Open Book —— 8086/88, 186/188 Instr, DOS & BIOS Services
NDP: none
Instruction   ADC          ADD WITH CARRY
1 µP INFO
2 µP INFO     Syntax:  ADC  destination, source
3 PSP INFO    Flags Affected:  CF  PF  AF  ZF  SF  OF
4 DOS INTS
5 DOS FUNC    Sums two byte or word operands adding one if CF set and re-
6 BIOS INTS   placing destination with result.

AAA
AAD
AAM
AAS           —— OPERANDS——————— CLOCKS——————TRANS——BYTES
ADC           register,register        3 (3)        -       2
ADD           register,memory          9 (10)+E     1       2-4
AND           memory,register          16 (10)+E    2       2-4
BOUNDs        register,immediate       4 (4)        -       3-4
CALL          memory,immediate         17 (16)+E    2       3-6
CBW           accumulator,immediate    4 (3-4)      -       2-3
CLC

↓↑ cursor RETURN selects  PgUp/PgDn browse  ALPHA KEY lookup  ESC exits
```

Sample Screen — 3 Manuals on 1 screen: BIOS, DOS and Instruction Sets

Look for Yourself!

OPEN BOOK	NORTON
Efficient 50K Memory Storage	Bulky 300K Disk Storage
Direct Lookup of Commands	Tree Structure Lookup
No Installation	Requires Configuration
5 Hot Keys	Fixed Keys
$79 for C and Assembler	$150 for C and Assembler
Eliminates 6 Manuals	Also Eliminates Manuals

$49 for One
NOW only $79 for Both
Intel 8086/88, 186/188 Instruction Set
Microsoft C Compiler Version 4.00

Call **415·435·0770**

Headlands

P.O. Box 8
Tiburon, California 94920

Microsoft & Intel are registered trademarks of Microsoft and Intel Corporation, respectively.

CIRCLE 354 ON READER SERVICE CARD

Listing One *(Text begins on page 18.)*

```
/*------------------------------------------------------------------
        main.c
        Program skeleton for testing the contour map algorithm

        This program reads in a MacPaint document, unpacks
        it, and calls the contour analysis algorithms.

        Once the analysis is complete the program writes out a
        file called grid.dat that can be read by hidlinpix and
        displayed in 3D perspective with hidden lines removed.

        hidlinpix is a program published in "Programming
        Principles in Computer Graphics" by Leendert
        Ammeraal (John Wiley & Sons, 1986)

        Other notes:

        Data structure libraries are derived from Alan Holub's
        column in Doctor Dobb's Journal.  These include the AVL
        tree and the queue.

        I modified the Lightspeed C stdio library so that the stdio
        console window occupies only the lower 1/3 of the screen.
        I then use the upper 2/3 for a window to display the source
        contour map and a second window to display the progress of the
        algorithm.

        William May
        303A Ridgefield Circle
        Clinton, MA 01510

        Jan 25, 1986        created
 ------------------------------------------------------------------*/

#include <stdio.h>
#include <WindowMgr.h>
#include "contour.h"

TREE                *root = 0L;
WindowPtr left_wind;
WindowPtr right_wind;
BitMap              bmap;
curve_data          curves[100];            /* array of curve data */

main()
{
        char c;
        Rect r;
        int width;

        /* force the LSC stdio window to open by doing a printf */
        printf("Program Starting\n\n");

        /* suppress the LSC dialog window when the program ends */
        Click_On(false);

        /* open some windows... */
        init_windows();

        init_mem();

        /* read in the MacPaint document, named "test" */
        if (read_MP("\ptest", &bmap)) {
                r.top     = 0;
                r.bottom  = VBITMAX;
                r.left    = 0;
                r.right   = HBITMAX;
                show_bmap(left_wind, &bmap, &r);

                find_all_contours();
        }

        make_hidlin();       /* make the input file for hidlinpix */

        /*
           A "real" program should return allocated memory to the
           system here, etc.  I will just return and let the heap
           reinitialization take care of everything.
        */

        printf("Program complete. <cr> to continue: ");
        c = getchar();
}

/*------------------------------------------------------------------
        init_windows opens the two windows used for testing graphics
        algorithms.  The console window is opened automatically
        by stdio when a printf is called.  No window template resources
        are used.  The two window records are kept in application global
        space.
 ------------------------------------------------------------------*/
init_windows()
{
        static WindowRecord left_wrec, right_wrec;
        int height, width;
        Rect r;

        height = 180;
        width  = 144;

        r.top    = 22;
        r.left   = 20;
        r.bottom = r.top + height;
        r.right  = r.left + width;

        left_wind = NewWindow(&left_wrec, &r, "\p", true, altDBoxProc,
                        FrontWindow(), false, 0L);

            r.left = 184;
            r.right = r.left + width;

            right_wind = NewWindow(&right_wrec, &r, "\p", true, altDBoxProc,
                        FrontWindow(), false, 0L);
    }
```

End Listing One

Listing Two

```
/*
            global.h

            contains the external references to global variables
*/

extern  TREE                    *root;              /* root of AVL tree */
extern  WindowPtr     left_wind;
extern  WindowPtr     right_wind;
extern  BitMap                  bmap;                              /* bitmap being
                                                                          used */
extern     curve_data           curves[100];             /* array of curve data */
```

End Listing Two

Listing Three

```
/* ----------------------------------------------------------------------
 *          contour.h
 *
 *          defines the leaf structure and does the usual includes
 * ---------------------------------------------------------------------- */

#include <WindowMgr.h>

typedef struct {
            union {
                        long key;               /* key = h,v concatenated */
                        Point p;
            } header;
            int  curve;             /* curve that this point is on */
} LEAF;

#define HBITMAX 576  /* max pixels in horiz direction */
#define VBITMAX 720 /* max pixels in vert direction */

#include "tree.h"

typedef struct curve_data {
            Point       p;                                   /* starting point of curve */
            int                         parent;              /* parent (enclosing) curve */
            int         elevation;                           /* elevation of the curve */
            int         searched;                            /* has interior been
                                                                           searched? */
} curve_data;
```

End Listing Three

Listing Four

```
/*----------------------------------------------------------------------
            tracer.c

            There is only one externally visible function: find_all_contours

            Implements contour search and tracing algorithms
            Some sections are based on Pavlidis, "Algorithms for Graphics and
            Image Processing"

            Quick description: the algorithm examines the interiors of known
            contours to find additional contours.

            It is started by treating the border of the image itself as a contour.

            The final outcome of this algorithm is a tree containing all points
            on contours in the image.  The key for a leaf in the tree is the point
            coordinates.  Each leaf also contains an index for the contour containing
            the point, and the elevation for the point.

            William May

            created             Jan 31, 1987
            modified  Apr 10, 1987          improve the trace logic to handle
                                contours that are multiple pixels in
                                thickness.
  ----------------------------------------------------------------------*/


#define DEBUG

#ifdef DEBUG
#define D(x)        x
#else
#define D(x)
#endif

#include <QuickDraw.h>
#include <MemoryMgr.h>
#include <stdio.h>
#include "getpixel.h"
#include "contour.h"
#include "global.h"

/* two defines for our trace directions */
#define CLOCKWISE -1
#define COUNTERCLOCKWISE 1

/* the index of the image borders is BASE (the exterior contour) */
#define BASE        -1

/* define the queue elements */
typedef struct {
            int h, v, chain;
} q_item;
```

(continued on next page)

DAN BRICKLIN'S DEMO PROGRAM ONLY $74.95

Read what they're saying about this popular program for prototyping and demo-making:

"A winner right out of the starting gate. After you use DEMO once, you'll wonder how you got along without it."
—PC Magazine

"Everybody who writes software, either commercially or for in-house applications, should immediately order a copy. Period. No exceptions."
—Soft·letter

Product of the Month
—PC Tech Journal

Thousands of developers and most of the largest and best known software companies are using this program. You can, too. Act now!

NEW TUTORIAL! JUST $49.95

The perfect companion to the Demo Program. The Tutorial helps you learn the ins and outs of its basic and advanced features. Complete with a 96 page manual containing step-by-step instructions, diskette, and function key template.

ORDER NOW! 1-800-CALL-800 x8088

VISA MasterCard AMERICAN EXPRESS

Use 800-number for orders only.
Questions, special shipping, etc., call **617-332-2240.**

No Purchase Orders. Massachusetts residents add 5% sales tax. Outside of the U.S.A., add $15.00.

Requires 256K IBM PC/Compatible, DOS 2.0 or later. Supports Monochrome, Color Graphics, and EGA Adapters (text mode only). The Tutorial requires the Demo Program.

SOFTWARE GARDEN, INC.
Dept. D
P.O. Box 373, Newton Highlands, MA 02161
CIRCLE 314 ON READER SERVICE CARD

TRUE
MULTITASKING
With
MultiDos Plus

"multitasking for the IBM-PC."

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in most languages.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
  - Intertask message communication. Send/receive/check message present on 64 message queues.
  - Task control by means of semaphores. Get/release/check semaphores.
  - Change priority-256 priority levels.
  - Suspend task for specified interval.
  - Spawn and terminate external and internal tasks.
  - Disable/enable multitasking.
  - and more!
- Independent foreground/background displays.
- Access to DOS while applications are running.

Hardware/Software Requirements

IBM PC/XT/AT or true clone. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

only: **$24.95** *OR*
$99.95
with source code

Outside USA add $5.00 shipping and handling.
Visa and Mastercard orders only call toll-free: 1-800-872-4566, ext. 350., or send check or money order to:

NANOSOFT
13 Westfield Rd, Natick, MA 01760
MA orders add 5% sales tax.

CIRCLE 309 ON READER SERVICE CARD

Listing Four *(Listing continued, text begins on page 18.)*

```
static char *qp;                        /* pointer to the queue */
static int          qmax = 0;           /* max items in the queue */

static int current_elevation = 0;       /* current elevation of region */
static int next_curve = 0;              /* number of next index in curve array */

static long point_count;                /* count the points we have found */

/*----------------------------------------------------------------
          find_all_contours finds all the contours in a bitmap
          first the left side of the bitmap is entered into the queue to
          start off the search.  then each time find_contours returns
          (meaning no more contours at that level) the elevation is incremented
          for the next round of searches.
  ---------------------------------------------------------------*/
int find_all_contours()
{
          char ch;
          Point pt;
          GrafPtr old_port;
          int draw_point();
          int item;
          register int i;
          char *makequeue();

          qp = makequeue(4000, sizeof(q_item));

          start_queue();                              /* set up search for exterior of bitmap */

          find_contours(BASE);                        /* find the first level of contours */
                                                      /* parent = -1 */

          while (next_point(&item, &pt)) {            /* while there is an unsearched curve */
                    trace(pt, COUNTERCLOCKWISE, true);  /* set up queue for next search */
                    find_contours(item);              /* go get 'em. item = parent */
          }

          free(qp);                                   /* all done, get rid of the queue */

          set_elevations();                           /* set elevations in the array */

          printf("max used in queue is %d\n", qmax);
          printf("number of curves found is %d\n", next_curve);
          printf("number of points found is %ld\n", point_count);
          printf("Curve search complete. <cr> to continue: ");
          ch = getchar();

          GetPort(&old_port);
          SetPort(right_wind);
          EraseRect(&(right_wind->portRect));
          D(printf("Now traversing the tree.\n"));
          D(traverse(root, draw_point));

          SetPort(old_port);
}

/*----------------------------------------------------------------
          start_queue puts the left edge of the bitmap into the
          queue to act as start for the bitmap search

          only the left edge needs to be put in because the other edges
          would be ignored anyway
  ---------------------------------------------------------------*/
static start_queue()
{
          register int i;
          q_item q;

          q.h = 0;
          q.chain = 6;                    /* all starting queue items are going south */

          for (i = 0; i < VBITMAX; i++) {
                    q.v = i;
                    if (!enqueue(&q,qp))
                              syserr(0, "Queue overflow in startup function\n");
          }
}

/*----------------------------------------------------------------
          set the next curve starting point if there is one,
          otherwise return 0;
  ---------------------------------------------------------------*/
static int next_point(item,pt)
int       *item;
Point     *pt;
{
          static int last_searched = -1; /* last array index used */

          if (++last_searched < next_curve) {
                    *item = last_searched;
                    pt->h = curves[last_searched].p.h;
                    pt->v = curves[last_searched].p.v;
                    return 1;
          }
          else
                    return 0;
}

/*----------------------------------------------------------------
          set the elevations in the array
          assume all contours are ascending
          and increment = 100
  ---------------------------------------------------------------*/
set_elevations()
{
          register curve_data *p = curves;
          register int parent, i;

          for (i = 0; i < next_curve; i++, p++) {
                    if ((parent = curves[i].parent) == BASE)
                              curves[i].elevation = 100;
                    else
                              curves[i].elevation = curves[parent].elevation + 100;
          }
}

/*----------------------------------------------------------------
          find_contours searches for all contours within a closed contour,
          based on the queue elements
          if any are found a non-zero value is returned
  ---------------------------------------------------------------*/
static int find_contours(parent)
int parent;                                       /* index of enclosing curve */
{
        q_item item, *show_next();
        register int c;
        Point p;
        int B, next_chain;
        Point first, last;

        while (dequeue(&item, qp)) {
                c = item.chain;

                if (c == 8) {
                        dequeue(&item, qp);
                        c = item.chain;
                }

                if ((c >= 5 && c <= 7) ||
                        (c == 0 && ((next_chain = (*show_next(qp)).chain) >= 5) &&
                        (next_chain <= 7))) {
                                p.h = item.h + 1;
                                p.v = item.v;

                                D(first = p);

                        do {
                                search_x(&B, &p);

                                if (B == 1) {

                                        D(last = p);
                                        D(show_line(first, last));

                                        /*
                                                if trace found a new contour then break off
                                                the scan.  If the point is a new point on a
                                                known contour, then continue.

                                        */
                                        if (trace(p, CLOCKWISE, false)) {
                                                /* and add curve to array */
                                                curves[next_curve].p = p;
                                                curves[next_curve].parent = parent;
                                                curves[next_curve].elevation = 0;
                                                curves[next_curve].searched = false;
                                                next_curve++;
                                                break;
                                        } else
                                                p.h++;    /* bump up to next point */
                                }
                                else if (B >= 2) {
                                        D(last = p);
                                        D(show_line(first, last));
                                        break;
                                }
                        } while (1);
                }
        }
        return 1;
}

/*----------------------------------------------------------------------
        scan pixels in the x direction at point p
        if the bitmap shows a pixel return 1
        if the bitmap shows a pixel and the pixel is in the tree
        (i.e. the point has been scanned before) return 2
        update the starting point
  ---------------------------------------------------------------------*/
static search_x(b, p)
int *b;
Point *p;
{
        long dcmp();
        /*
                the union makes it easy to use the point coordinates
                as the key in the tree
        */
        union {
                Point pt;
                long  key;
        } q.

        q.pt = *p;

        if (*b = getpixel(q.pt.h, q.pt.v, &bmap)) {
                if (find(root, q.key, dcmp)) (*b)++;
                return;                                 /* don't increment h on a hit */
        }

        if (++p->h >= HBITMAX) {
                *b = 2;                         /* when we hit the right edge,indicate already
                                                                                        found */

                return;
        }
}

/*----------------------------------------------------------------------
        trace traces a contour beginning at the specified point
        at each found point the x,y coordinates and chain code are
        stored both in the queue and the tree.

        two result codes are returned:
        0 indicates that the starting point represented a new point on an
          existing (known) contour.  trace was aborted.
        1 indicates a new contour was successfully traced
  ---------------------------------------------------------------------*/
static int trace(p, dir, q_only)
Point p;                /* p is the starting point */
int dir;                /* dir is the trace direction */
Boolean q_only; /*  trace and put in queue only,
                                                do not check for dupes in tree */
{
        Point c, t;             /* current point, and transformed point for testing */
        int s;      /* search direction */
        Boolean found, test_pixel();
        int count, used, chain;
        q_item item;
        LEAF *lp, *lp2;
        long icmp(), dcmp();
        GrafPtr old_port;
        union temp {
                long key;
                Point p;
        } temp;

        if (dir == COUNTERCLOCKWISE)
                s = 6;
        else
                s = 2;

        c = p;
```

(continued on next page)

Dbase*

programming tools

*Clipper, FoxBASE+, dBASE, QuickSilver

The UI Programmer

UI is the first professional code generator; we wrote UI for programmers who want to automate their work but cannot use code that is 'almost' good enough. If your user interfaces include bounce-bar menus, pop-up help screens and the other features of today's best programs, you will gain an order of magnitude in productivity with UI.

UI is a second generation, programmable product — so your code comes out your way. Application specific edits, for instance, can be placed in the UI 'template' which controls the generation. Edit the screen appearance until it 'looks and feels' perfect. Everytime you generate code, your special logic is preserved.

Speaking of editing the screen, UI includes a powerful, 3-D screen editor, so you can draw pop-up help boxes over your pull-down menus, over your application.

The Documentor

To run Doc, you just tell it the name of the main-line routine and make sure your printer has a lot of paper! (Sure, you can have the output go to the screen or a file, too.)

You can tailor your documentation to include any or all of: a table of contents, system tree diagram (main line is the root), hierarchy (box diagram) charts for each module, action diagrams (modern style flow charts) for each PRG or procedure, DBF listings (structure, indexes, more), where used/updated listings for fields and all variables — by module and by line number within each module.

Our written money-back satisfaction guarantee set a new standard when we began it in 1985. (Return rate to date: 0.6% and dropping!) No copy protection, royalties or other nonsense.

Suggested retail: $295 each, (800) support included. At your dealer today. Call us for a very special offer on our latest release! (800) 233-3569 or, in NY, (212) 406-7026.

WallSoft

The Computer Aided Software Engineering Corporation
233 Broadway, Suite 869, New York, NY 10279

CIRCLE 90 ON READER SERVICE CARD

A Reconfigurable Programmer's Editor With Source Code

ME is a high quality programmer's text editor written specifically for the IBM PC. It contains features only found in the more expensive programmer's text editors. These features include:

- **Multiple Windows**
- **Column cut and paste**
- **Line marking for source code**
- **Regular Expressions**
- **C-like Macro Language**
- **Reconfigurable Keyboard**
- **Capture your DOS session**
- **Run your compiler and examine errors**
- **Comes with useful precompiled macros**
- **UNIX-like CTAGS**
- **43-line mode with EGA**

This is the **ONLY** editor with the power of the higher priced editors which comes with complete source code!

New commands and features may be added to the editor by writing programs in its macro language. The language resembles C, and comes with a rich set of primitives for handling strings and changing the editor environment, plus most of the flow-of-control constructs that come in C *(for, while, if, break, continue)*.

The code is written in standard C, with several key library routines written in assembler for speed. The source code option is perfect for OEMs and VARs who want to add editing or word processing capabilities to their applications.

Price for editor and on-line documentation ---- $39.95

Price for editor with complete source code -- $94.95

(Please add $2 for shipping & handling, $6 overseas)

Special offer -- New York Word word processor -- $39.95 -- Multi-windowing, mail merge, hyphenation, math, regular expressions, TOC and index generators

MAGMA SYSTEMS

138-23 Hoover Ave. Jamaica, NY 11435 (201) 792 - 3954

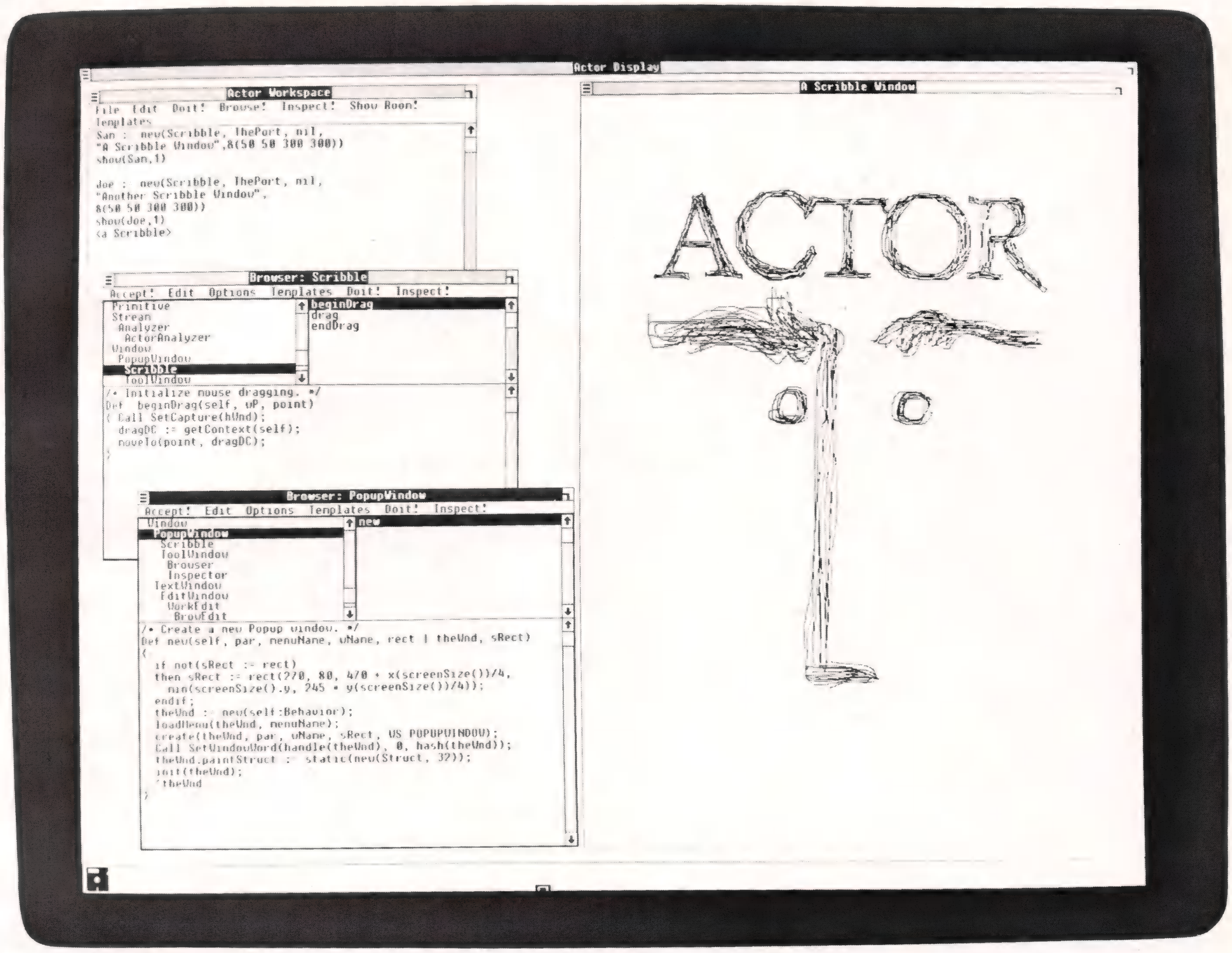
CIRCLE 313 ON READER SERVICE CARD

Listing Four *(Listing continued, text begins on page 18.)*

```
if (q_only) {
        /* has this curve interior been searched before? */
        temp.p.h = c.h;
        temp.p.v = c.v;
        lp = find(root, temp.key, dcmp);

        if (curves[lp->curve].searched == true) {
                printf("Curve %d already searched\n", lp->curve);
                return 0;
        }
        else
                curves[lp->curve].searched = true;

        item.h = c.h;           /* add data to queue */
        item.v = c.v;
        item.chain = 8;         /* mark beginning of new contour */
        if (!enqueue(&item,qp))
                syserr(0, "Queue overflow in trace\n");

        GetPort(&old_port);
        SetPort(right_wind);
        PenPat(white);
        SetPort(old_port);
}
else {
        if (!(lp = talloc(sizeof(LEAF))))
                syserr(MemErr, "Insufficient memory for AVL tree");
        else {
                lp->header.p.h = c.h;
                lp->header.p.v = c.v;
                lp->curve = next_curve;

                if (lp2 = insert(&root, lp, icmp)) {
                        /* tfree(lp); not implemented in getmem! */
                        return 0; /* starting point already in the tree !! */
                }
                else {  /* if not in the tree then enqueue the new point */
                        point_count++;

                        item.h = c.h;           /* add data to queue */
                        item.v = c.v;
                        item.chain = 8;         /* mark beginning of new contour */
                        if (!enqueue(&item,qp))
                                syserr(0, "Queue overflow in trace\n");

                        GetPort(&old_port);
                        SetPort(right_wind);
                        PenPat(black);
                        SetPort(old_port);
                }
        }
}


do {
        found = false;
        count = 0;
        while (!found && (count < 3)) {
                count++;
                if (test_pixel(c, add_chain(s, -1 * dir))) {
                        set_pixel(&c, (chain = add_chain(s, -1 * dir)));
                        s = add_chain(s, -2 * dir);
                        found = true;
                }
                else {
                        if (test_pixel(c, s)) {
                                set_pixel(&c, (chain = s));
                                found = true;
                        }
                        else {
                                if (test_pixel(c, add_chain(s, 1 * dir))) {
                                        set_pixel(&c, (chain = add_chain(s, 1 * dir)));
                                        found = true;
                                }
                                else
                                        s = add_chain(s, 2 * dir);
                        }
                }
        }
        D(show_point(c));

        if (q_only) {
                item.h = c.h;           /* add data to queue */
                item.v = c.v;
                item.chain = chain;
                if (!enqueue(&item,qp))
                        syserr(0, "Queue overflow in trace\n");
        }
        else {
                if (!(lp = talloc(sizeof(LEAF))))
                        syserr(MemErr, "Insufficient memory for AVL tree");
                else {
                        lp->header.p.h = c.h;
                        lp->header.p.v = c.v;
                        lp->curve = next_curve;

                        if (lp2 = insert(&root, lp, icmp)) {
                                /* tfree(lp); not implemented in getmem! */
                                if ((c.h == p.h) && (c.v == p.v))
                                        return 1; /* we have returned to the start */
                                else
                                        return 0;
                        }
                        else {                          /* if not in the tree then enqueue point */
                                point_count++;

                                item.h = c.h;           /* add data to queue */
                                item.v = c.v;
                                item.chain = chain;
                                if (!enqueue(&item,qp))
                                        syserr(0, "Queue overflow in trace\n");
                        }
                }
        }
        if ((used = sp_used(qp)) > qmax) qmax = used;
} while ((c.h != p.h) || (c.v != p.v));

GetPort(&old_port);
SetPort(right_wind);
PenPat(black);
SetPort(old_port);
        return 1;
}
```

(continued on page 72)

HOW TO WRITE A WINDOWS APPLICATION IN TEN MINUTES.



```
Sam := new(Scribble, ThePort, nil,
"A Scribble Window",&(50 50 300 300))
show(Sam,1)

Joe := new(Scribble, ThePort, nil,
"Another Scribble Window",
&(50 50 300 300))
show(Joe,1)
<a Scribble>
```

```
/* Initialize mouse dragging. */
Def  beginDrag(self, wP, point)
{ Call SetCapture(hWnd);
  dragDC := getContext(self);
  moveTo(point, dragDC);
}
```

```
/* Create a new Popup window. */
Def new(self, par, menuName, wName, rect | theWnd, sRect)
{
  if not(sRect := rect)
  then sRect := rect(270, 80, 470 + x(screenSize())/4,
    min(screenSize().y, 245 + y(screenSize())/4));
  endif;
  theWnd := new(self:Behavior);
  loadMenu(theWnd, menuName);
  create(theWnd, par, wName, sRect, WS_POPUPWINDOW);
  Call SetWindowWord(handle(theWnd), 0, hash(theWnd));
  theWnd.paintStruct := static(new(Struct, 32));
  init(theWnd);
  ^theWnd
}
```

Actor™ is a new language that combines Microsoft® Windows with object-oriented programming. This means you can produce mouse and window applications very quickly.

For example, we created a simple "paint" program, and used it to draw the Actor logo you see on the screen. The whole program only took ten lines and ten minutes. Part of it is in the middle window on the left.

Above, you see the commands that initialized the paint window and made it appear on the screen. Below, some code that's built into Actor, specifying window behavior. Through a process known as "inheritance," it's called into play automatically.

Try programming in this new way, and you'll never go back.

Find out about Actor.
Call The Whitewater Group,® (312) 491-2370.

Technology Innovation Center
906 University Place, Evanston, IL 60201

MICROSOFT WINDOWS

CIRCLE 282 ON READER SERVICE CARD

Listing Four

(Listing continued, text begins on page 18.)

```
/*---------------------------------------------------------------------
          test the s neighbor of point p
  ---------------------------------------------------------------------*/
static Boolean test_pixel(p,s)
Point p;
int s;   /* direction to test */
{
          set_pixel(&p, s);
          return (getpixel(p.h, p.v, &bmap));
}

/*---------------------------------------------------------------------
          set point p to point + chain code s
  ---------------------------------------------------------------------*/

static set_pixel(p, s)
Point *p;
int s;
{
          switch (s) {
                    case 0:
                              (*p).h++;
                              break;
                    case 1:
                              (*p).h++;
                              (*p).v--;
                              break;
                    case 2:
                              (*p).v--;
                              break;
                    case 3:
                              (*p).h--;
                              (*p).v--;
                              break;
                    case 4:
                              (*p).h--;
                              break;
                    case 5:
                              (*p).h--;
                              (*p).v++;
                              break;
                    case 6:
                              (*p).v++;
                              break;
                    case 7:
                              (*p).h++;
                              (*p).v++;
                              break;
          }
}

/*---------------------------------------------------------------------
          add c to chain code s
  ---------------------------------------------------------------------*/
static int add_chain(s,c)
register int s,c;
{
          register int i;

          if ((i = s + c) >= 8)
                    return (i - 8);
          else {
                    if (i < 0)
                              return (i + 8);
                    else
                              return i;
          }
}

#ifdef DEBUG

/*---------------------------------------------------------------------
          The code below is used to display the progress of the contour
          analysis in a Macintosh window.  It is obviously totally
          nonportable.  Fixed point arithmetic is used for scaling the MacPaint
          size document to a much smaller window.  Fixed point is much faster
          floating point, and is often used for two and three dimensional
          graphics on the Mac.  This code is for debugging purposes.
  ---------------------------------------------------------------------*/

/* ratio is used to scale down picture size by 1/4 */
static Fixed ratio = 0x00004000;

/*---------------------------------------------------------------------
          convert a 16 bit integer to fixed point
  ---------------------------------------------------------------------*/
static Fixed int_to_fix(i)
int i;
{
          asm {
                    move.w    i,d0        ; get the number
                    ext.l     d0          ; clear top of d0
                    swap      d0          ; make it a fixed point number,
                                                                  all done
          }
}

/*---------------------------------------------------------------------
          convert a Fixed point number to 16 bit integer
  ---------------------------------------------------------------------*/
static int fix_to_int(i)
Fixed i;
{
          asm {
                    move.l    i,d0        ; get the number
                    add.l     #0xA000,d0; add .5 to number, for rounding
                    swap      d0          ; make the int, ignore upper word of d0

            }
  }

  /*---------------------------------------------------------------------
            draw the point from a LEAF, scaled down to the actual window size
    ---------------------------------------------------------------------*/
  static draw_point(p)
  LEAF *p;
  {
            GrafPtr oldPort;
            Point q;
            Rect r;

            q = p->header.p;

            GetPort(&oldPort);
            SetPort(right_wind);

            r.top    = fix_to_int(FixMul(int_to_fix(q.v), ratio));
            r.left   = fix_to_int(FixMul(int_to_fix(q.h), ratio));
            r.right  = r.left + 1;
            r.bottom = r.top + 1;

            FrameRect(&r);
            SetPort(oldPort);
  }

  /*---------------------------------------------------------------------
            draw the point, scaled down to the actual window size
    ---------------------------------------------------------------------*/
  static show_point(p)
  Point p;
  {
            GrafPtr oldPort;
            Rect r;

            GetPort(&oldPort);
            SetPort(right_wind);

            r.top    = fix_to_int(FixMul(int_to_fix(p.v), ratio));
            r.left   = fix_to_int(FixMul(int_to_fix(p.h), ratio));
            r.right  = r.left + 1;
            r.bottom = r.top + 1;

            FrameRect(&r);
            SetPort(oldPort);
  }

  /*---------------------------------------------------------------------
            draw a line, scaled down to the actual window size
    ---------------------------------------------------------------------*/
  static show_line(first, last)
  Point first,last;
  {
            GrafPtr oldPort;

            GetPort(&oldPort);
            SetPort(right_wind);

            first.v  = fix_to_int(FixMul(int_to_fix(first.v), ratio));
            first.h  = fix_to_int(FixMul(int_to_fix(first.h), ratio));

            MoveTo(first.h, first.v);

            last.v  = fix_to_int(FixMul(int_to_fix(last.v), ratio));
            last.h  = fix_to_int(FixMul(int_to_fix(last.h), ratio));

            LineTo(last.h, last.v);

            SetPort(oldPort);
  }


  #endif
```

End Listing Four

Listing Five

```
/*
          stack.h

          William May
          2/20/87                 created
*/

typedef int *STACK;

/* function prototypes */
extern int          empty       ( STACK * );
                                        /* is anything on the stack? */

extern int          pop         ( STACK * );
extern int          push        ( int, STACK *);
extern STACK *      init_stack( int );
extern void         del_stack   ( STACK ** );        /* remove stack */
extern int                      top_of_stack( STACK * );
                                             /* returns top of stack */
```

End Listing Five

Listing Six

```
/*-----------------------------------------------------------------
          stack.c

          implements a simple stack of integers

          William May
          303A Ridgefield Circle
          Clinton, MA 01510

          2/20/87                 created
  -----------------------------------------------------------------*/

#ifdef DEBUG
#include <stdio.h>
#endif

#include <storage.h>
```

```
typedef struct STACK {
          int max;              /* max items in stack */
          int top;              /* current items in stack */
          int items[];          /* the data */
} STACK;

/* some error codes */
#define FULL  -2
#define EMPTY -1
#define NOERR  0

#ifdef DEBUG

STACK *mystack;

main()
{
          STACK *init_stack();
          int num, c;

          printf("Stack program begun\n");

          if (!(mystack = init_stack(5))) {
                    fprintf(stderr, "Insufficient memory to create stack\n");
                    ExitToShell();
          }

          while( 1 ) {
                    num = c = -1;

                    printf("top              = %d\n", mystack->top);
                    printf("max              = %d\n", mystack->max);
                    printf("top item = %d\n", top_of_stack(mystack));


                    printf("\n<p(op)/s(push)/q(uit)> -> ");
                    while( c != 'p' && c != 's' && c != 'q' )
                              c = getchar();

                    if( c == 's' )
                    {
                              printf("\nenter decimal number -> ");
                              scanf("%d", &num );
                              printf("\npush(%d) returned %d\n",
                                                            num, push(num, mystack));
                    }
                    else if( c == 'p' )
                    {
                              printf( "\npop returned: %d\n", pop(mystack));
                    }
                    else
                              break;
          }

          del_stack(&mystack);

          printf("\nStack program complete\n");
}
#endif

/*-------------------------------------------------------------------
          create the stack
          NULL returned if insufficient memory
 ------------------------------------------------------------------*/
STACK *init_stack    ( items )
int items;
{
          STACK * p;

          if (p = (STACK *)calloc ((sizeof(int) * items + sizeof(STACK)),1)) {
                    p->max = items;
                    p->top = EMPTY;
          }

          return (p);
}

/*-------------------------------------------------------------------
          delete the stack
 ------------------------------------------------------------------*/
void del_stack ( stackptr )
STACK **stackptr;
{
          free(*stackptr);
          *stackptr = 0L;
}

/*-------------------------------------------------------------------
          is the stack empty?
 ------------------------------------------------------------------*/
int empty ( stack )
STACK *stack;
{
          return ((stack->top == EMPTY) ? 1 : 0);
}

/*-------------------------------------------------------------------
          return the top element of the stack and decrement
          the stack pointer
 ------------------------------------------------------------------*/
int pop ( stack )
STACK *stack;
{
          if (empty(stack))
                    return EMPTY;
          else {
                    return (stack->items[stack->top--]);
          }
}
```

(continued on next page.)

goodbye dBase!

dBASE Programmers

You need it!
You can handle it!
dB2c is here now!

dB2c Offers:

- Version 2.0 complete with Translator and File Handlers.
- Extensive implementation of dBASE III+ with over 200 functions and commands in C source code.
- Contains our own File Handlers plus interfaces for Lattice's dBC and Faircom's c-tree.
- Supports screen I/O with ANSI.SYS or fast assembler routines.
- Support for Microsoft, Lattice and Turbo C compilers.
- Tutor features of translation combined with familiar syntax of the library eases the transition to 'C'.
- One version supports MS-DOS, Xenix, Unix, OS-9 and Concurrent DOS.

are you ready?

dB→c

Toolkit $299

DAVID J.
MARSH

Call or Write:
SOFTWARE CONNECTION, INC.
POB 712, Ely, MN 55731
(218) 365-5097

CIRCLE 330 ON READER SERVICE CARD

Listing Six *(Listing continued, text begins on page 18.)*

```
/*-------------------------------------------------------------------
          push an integer onto the stack and increment the stack pointer
  -------------------------------------------------------------------*/
int push (n, stack)
int n;
STACK *stack;
{
          if (++stack->top < stack->max) {
                    stack->items[stack->top] = n;
                    return NOERR;
          }
          else {
                    --stack->top;
                    return FULL;
          }
}

/*-------------------------------------------------------------------
          look at the top of the stack without changing the
          stack pointer
  -------------------------------------------------------------------*/
int top_of_stack( stack )
STACK *stack;
{
          return (stack->items[stack->top]);
}
```

End Listing Six

Listing Seven

```
/*
          make_hidlin.h

          header file for make_hidlin

          William May

          created:  3/21/87
*/

#define MAPWIDTH     576
#define MAPHEIGHT    720
#define INTERVAL     24        /* interval, in pixels, between grid lines */

#define VMAX 31                /* bitmap height/interval + 1 */
#define HMAX 25                /* bitmap width/interval + 1  */
```

End Listing Seven

Listing Eight

```
/*----------------------------------------------------------------------
          make_hidlin.c

          Creates a grid of elevations and then
          input file for the 3D hidden line algorithm (hidlinpix)

          The input file creation process is fully explained in Ammeraal's
          book "Programming Principle in Computer Graphics"

          William May
          303A Ridgefield Cirlce
          Clinton, MA 01510

          created:  3/20/87
  ---------------------------------------------------------------------*/

#include <stdio.h>
#include "make_hidlin.h"
#include "contour.h"

#define SCALE 100.0

int grid[HMAX+1][VMAX+1];
FILE *fp;

/*----------------------------------------------------------------------
          Coordinate creating an input file for hidlinpix
  ---------------------------------------------------------------------*/
make_hidlin()
{
          printf("Beginning make_hidlin\n");

          init_grid();

          make_grid();

          fp = fopen("grid.dat", "w");

          /* convert grid to hidlin input format */
          conv_hidlin();

          fclose(fp);
}

/*----------------------------------------------------------------------
          0 the grid
          this may be unnecessary for a global, but doesn't hurt
          It is however, necessary if the contour analysis is
          done repeatedly without quitting the program
  ---------------------------------------------------------------------*/
init_grid()
```

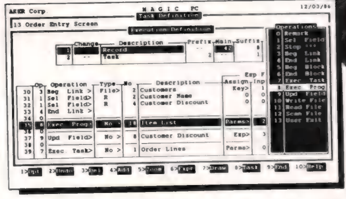
(continued on page 76

Make $500/hr.

Now develop DBMS applications 10 times faster for only $199 with MAGIC PC — or your money back!

Database programmers, why waste your time hacking out code?
Imagine how much faster and more profitable you'd be if you could whip up powerful database applications without the time-consuming coding pains...
Introducing Magic PC from Aker, your professional dream come true. It's not another line-by-line syntax treadmill like any DBMS or 4GL.
Finally you can program as quickly as you design, while you delegate all the mundane and redundant coding tasks to Magic PC.

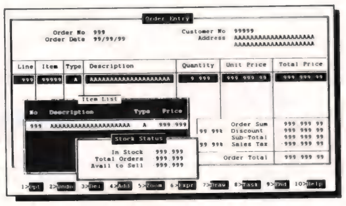
Program 10 times faster



```
AKER Corp.                     M A G I C   PC                          12/03/86
                              Task Definition
13 Order Entry Screen
                           Execution Definition                        Operations
          Change── Description ──────────Prefix─Main─Suffix            0 Remark
       1         Record                     --    42      0            1 Sel  Field
       2    --   Task                        2    --      1            2 Stop ***
                                                                       3 Beg  Link
                                                                       4 End  Link
                                                                       5 Beg  Block
                                                          Exp F        6 End  Block
    Op  Operation  Type  No      Description          Assign Inp       7 Exec  Task
30   3 Beg  Link > File>  2  Customers                  Key>   1       8 Exec  Prog
31   1 Sel  Field>  R     2  Customer Name                0    0       9 Upd  Field
32   1 Sel  Field>  R     4  Customer Discount            0    0      10 Write File
33   4 End  Link >                                                    11 Read File
34   0                                                                12 Scan File
35   8 Exec  Prog>  No >  18 Item List                Parms>   2      13 User Exit
36   0
37   9 Upd  Field>  No >   8 Customer Discount          Exp>   3
38   0
39   7 Exec  Task>  No >   1 Order Lines              Parms>   0

 1>Opt  2>Undo  3>Del  4>Add  5>Zoom  6>Expr  7>Draw  8>Task  9>End  10>Help
```

Develop relational database applications 10 times faster using a visual design-driven interface. Instead of writing mountains of "how to" procedural code, you quickly place your program design specs in Execution Tables and Magic PC's engine executes them automatically. Don't lose any more time editing and debugging programs by hand.

Incredible Zoom power



```
                              Order Entry
        Order No   999                     Customer No  99999
      Order Date   99/99/99                    Address  AAAAAAAAAAAAAAAAAAAA
                                                        AAAAAAAAAAAAAAAAAAAA
Line  Item   Type  Description            Quantity  Unit Price   Total Price
 999  99999   A    AAAAAAAAAAAAAAAAAAAA     9 999   999 999 99    999 999 99
               Item List
  No   Description             Type   Price
  999  AAAAAAAAAAAAAAAAAAAA     A    999 999          Order Sum   999 999 99
                                            99 99%   Discount    999 999 99
                    Stock Status                     Sub-Total   999 999 99
                                            99 99%   Sales Tax   999 999 99
               In Stock       999 999
           Total Orders       999 999               Order Total  999 999 99
          Avail to Sell       999 999

 1>Opt  2>Undo  3>Del  4>Add  5>Zoom  6>Expr  7>Draw  8>Task  9>End  10>Help
```

Magic PC's phenomenal Zoom power magically co-executes related programs through nested Zoom windows smoothly with auto data scrolling in all directions. While Zooming, query and transfer data across windows or even Zoom deeper.

No more maintenance!

Change your programs on the fly without any manual maintenance responsibility. Magic PC automatically updates your changes online since all the data describing your design (data dictionary, programs and menus) make up a single file, self-maintaining Integrated Library.

Magic PC does it all

Design your entire database application with only one comprehensive development system. Generate both online programs (screens, windows, menus), as well as batch programs (reports, updates, import/export, etc.) with full color and graphics. You no longer fall between the cracks dealing with separate and inconsistent programming utilities.

Free LAN features

Develop multi-user applications for local area networks with Magic PC's automatic support for file and record locking security.

Quick prototyping

Prototype a complete working application in just hours and get immediate customer feedback to finalize the design. It's a true time-saver.

Stand-alone runtime

Distribute your applications and protect your design with a low cost runtime engine. It has the friendliest end-user visual interface you've ever seen with built-in, menu-driven and syntax-free data retrieval power.

Jeff Duntemann, PC Tech Journal:

"*Magic PC is probably the best integrated database application generator that we have seen... very smooth system, and smoothness comes at a premium these days.*" Also recommended by PC Magazine, PC World, PC Week, Computer Language, Data Based Advisor and many more around the world.

Try it for $19⁹⁵

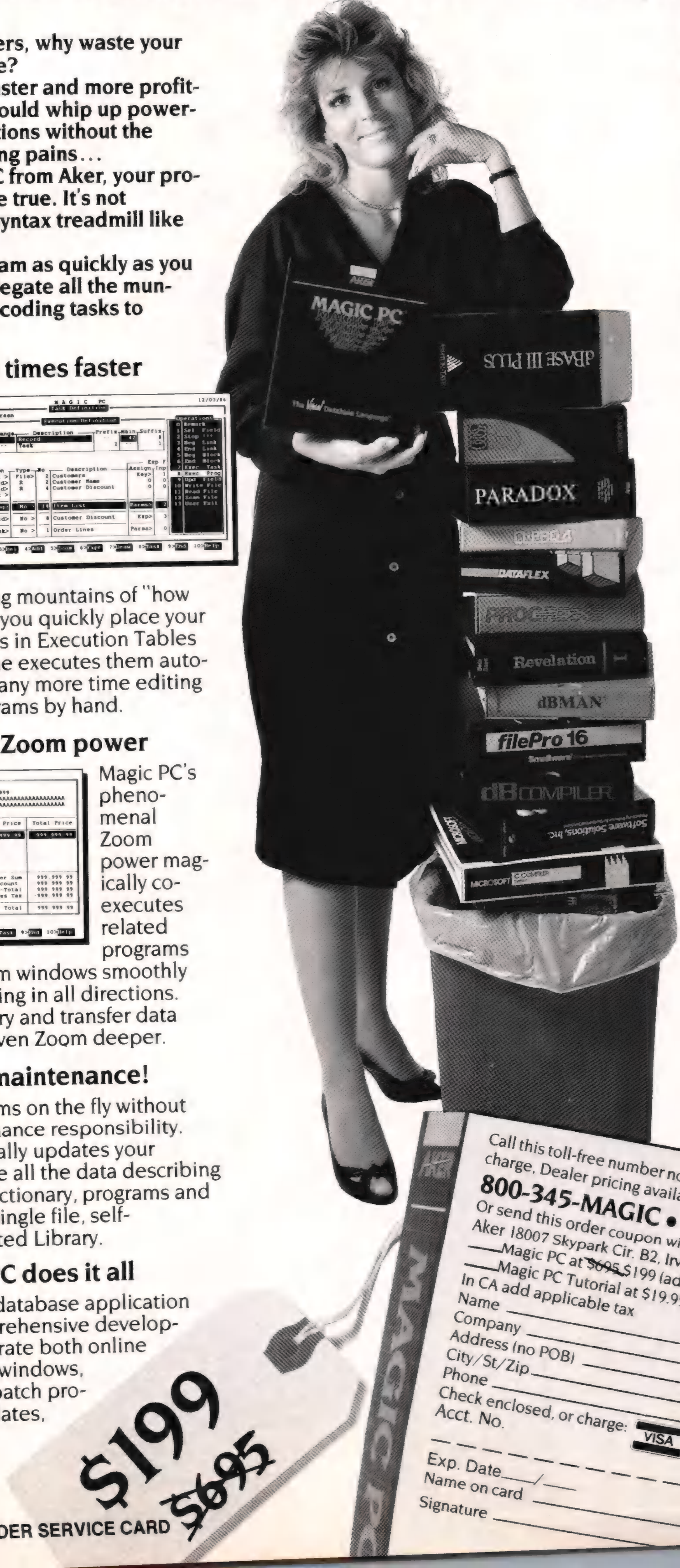
If you develop database applications for a living, you can't afford not to try Magic PC for yourself right now. For $19.95 you'll get the Magic PC Tutorial software and documentation for hands-on evaluation, complete with a step-by-step guide to develop an Order Entry sample application in just a few hours.

Magic PC ~~$695~~ $199

No kidding! For a limited time only, save almost $500 off the $695 list price, and get the complete unprotected Magic PC software for only $199 at our special introductory non-resale price.

Money back guarantee

Even at $199 you can't go wrong with our no-risk guarantee: keep it only if it makes magic for you, or we'll buy it back within 30 days less $19.95 restocking fee.



Call this toll-free number now with your credit card or COD charge. Dealer pricing available.
800-345-MAGIC • in CA 714-250-1718
Or send this order coupon with your payment to:
Aker 18007 Skypark Cir. B2, Irvine, CA 92714
——Magic PC at ~~$695~~ $199 (add $10 P&H)
——Magic PC Tutorial at $19.95 (add $5 P&H)
In CA add applicable tax

Name	
Company	1 2 3 4 5 6 7 8 9 0
Address (no POB)	
City/St/Zip	
Phone	
Check enclosed, or charge:	VISA MasterCard AMERICAN EXPRESS
Acct. No.	
Exp. Date	/
Name on card	
Signature	

System Requirements: IBM PC, XT, AT, PS/2 and 100% compatible, PC-DOS 2.0 or later, 512K, hard disk. All trademarks acknowledged.

MAGIC PC
The Visual Database Language
by AKER

CIRCLE 369 ON READER SERVICE CARD

MetaWINDOW

Power Graphics for your PC!

PC TECH JOURNAL

"Product of the Month"
"... a technological tour de force for fast PC graphics."

NO ROYALTIES!

MetaWINDOW is an advanced, high performance graphics development toolkit which bridges the gap between low-level graphic primitive libraries and pre-packaged window managers.

Unparalleled Performance!

MetaWINDOW provides an expanded set of graphic drawing functions, plus the added functionality and performance required for designing multi-window desktop applications.

- auto-cursor tracking
- pull-down menus
- pop-up windows
- comprehensive graphic functions
- multiple fonts

File
New ^N
Open
Close
Save ^S
Quit ^Q

10 Point 12 Point
Bold *Italic*

Enhanced Features!

- Display multiple bitmap or "filled-outline" fonts.
- Face fonts for bold, italic, underline or strike-out stylings.
- Full "RasterOp" transfer functions for writing, erasing, rubberbanding or dragging: lines, text, icons, bit images and complex objects.
- Create pop-up menus, windows and icons.
- Supports IBM's new PS/2 VGA and MCGA graphics.

MetaWINDOW comes complete with langauge bindings for 16 popular C, Pascal and Fortran compilers, plus dynamic runtime support for over 40 graphics adaptors and input devices.

MetaWINDOW
Advanced Graphics Toolkit
*4 disks, 3 260 page manuals - $195**

NEW! - TurboWINDOW/C
*All the features of MetaWINDOW for Borland Turbo C! - $95**
** Plus $5.00 shipping and handling*

TO ORDER CALL 1-800-332-1550
For information or in CA call 408-438-1550

METAGRAPHICS
SOFTWARE CORPORATION
269 Mount Hermon Road
Scotts Valley, CA 95066

CIRCLE 392 ON READER SERVICE CARD

Listing Eight *(Listing continued, text begins on page 18.)*

```
{
        register int h,v;

        for (h = 0; h <= HMAX; h++)
                for (v=0; v <= VMAX; v++)
                        grid[h][v] = 0;
}

/*--------------------------------------------------------------------
        Convert the grid to a hidlinpix input file
        This function is based on code from Ammeraal's book for
        doing 3D projections of mathematical functions
 --------------------------------------------------------------------*/
conv_hidlin()
{
        int i, j, k, l;
        double f();

        fprintf(fp, "%lf %lf %lf\n", (double)HMAX / 2.0, (double)VMAX / 2.0, 0.0);

        printf("Printing the point coordinates\n");

        for (i = 0; i <= HMAX; i++) {
                for (j = 0; j <= VMAX; j++) {
                        fprintf(fp, "%d %lf %lf %lf\n", j*(HMAX+1)+i+1,
                                (double)i, (double)j, f(i,j));
                }
        }

        fprintf(fp, "Faces:\n");

        printf("Printing the faces\n");

        /* next two lines switched */
        for (i = 0; i < HMAX; i++) {
                for (j = 0; j < VMAX; j++) {
                        k = j*(HMAX+1)+i+1;
                        l = k+HMAX+1;
                        fprintf(fp, "%d %d %d#\n", k, -(l+1), k+1);
                        fprintf(fp, "%d %d %d#\n", k+1, l+1, -k);
                        fprintf(fp, "%d %d %d#\n", k, -(l+1), l);
                        fprintf(fp, "%d %d %d#\n", l, l+1, -k);
                }
        }
}

/*--------------------------------------------------------------------
        note reversing the h coordinates
        somewhere the grid is being reversed, so I unreverse it here.
 --------------------------------------------------------------------*/
double f(h,v)
int h,v;
{
                . return ((double)(grid[HMAX - h][v]) / SCALE);
}
```

End Listing Eight

Listing Nine

```
/*-------------------------------------------------------------
        make_grid

        converts graph data into a grid that can be interpreted
        by hidlinpix

        William May
        303A Ridgefield Circle
        Clinton, MA 01510

        Feb 16, 1986                    created
 -------------------------------------------------------------*/

#define DEBUG

#ifdef DEBUG
#define P(x) x
#else
#define P(x)
#endif

#include <stdio.h>
#include "stack.h"
#include "make_hidlin.h"
#include "contour.h"
#include "global.h"

extern int stack_error;
extern int grid[HMAX][VMAX];

static STACK *stack;

/*--------------------------------------------------------------
        move vertically
        assigning elevations to each grid point
 --------------------------------------------------------------*/
make_grid()
{
        register int h;

        stack = init_stack(2000);       /* should be plenty! */

        for (h = 1; h < (HMAX-1); h++) {
                traverse_vert(h);
        }

        del_stack(&stack);
}
```

```
/*-----------------------------------------------------------------------
        Traverse a vertical grid line, determining elevations along the way
        This is mostly simple: i.e. keep track of the elevation of the
        contour line crossed last.
        The main complications are tangents (in which case we want to ignore
        the contour line) and inflection points (in which case we don't
        ignore the contour line).
  ----------------------------------------------------------------------*/
traverse_vert(h)
register int h;
{
        register int ver, hor;
        register int vmax = (VMAX - 1) * INTERVAL;

        hor = h * INTERVAL;

        push(0, stack);                 /* push 0 (elev of border) onto the stack */

        P(show_line(hor,1,hor,vmax));

        for (ver = 1; ver < vmax; ver++) {
                /* traverse a vertical grid line */
                /* on a grid intersection set array to value on the stack */
                if (!(ver % INTERVAL)) {
                        P(show_point(hor-3,ver));
                        P(show_point(hor+3,ver));
                        set_elevation(h, (ver / INTERVAL));
                }

                /* are we crossing a contour? are we tangent to it? */
                if (getpixel(hor, ver, &bmap)) {
                        if (is_crossing(hor, ver) || start_inflection(hor, ver))
                                ver += check_hit(hor,ver);
                }
        }
}
/*-----------------------------------------------------------------------
        set grid[h][v] to the value of top of stack
  ----------------------------------------------------------------------*/
static set_elevation(h, v)
register int h, v;   /* note these are grid coordinates !! */
{
        grid[h][v] = top_of_stack(stack);
}
/*-----------------------------------------------------------------------
        a pixel was hit, get the curve index from tree
        get curve elevation from curve array
        if curve elevation = elevation on stack
                pop stack
        else
                push new elevation
        end
        return the number of pixels to skip-1 (i.e.
        return 0 if wskip 1, etc.)
  ----------------------------------------------------------------------*/
static short check_hit(h, v)
register int h, v;   /* note these are pixel coordinates !! */
{
        short     cnt = 1;   /* number of pixels */
        short   ver;

        union temp {
                long key;
                Point p;
        } temp;
        LEAF *lp;
        long dcmp();
        int elev;

        /* traverse all pixels, if more than one */
        for (ver = v+1; getpixel(h, ver++, &bmap); cnt++)
                ;

        temp.p.h = h;
        temp.p.v = v;
        lp = find(root, temp.key, dcmp);

        elev = curves[lp->curve].elevation;

        if (elev == top_of_stack(stack))
                pop(stack);
        else
                push(elev, stack);

        return (cnt-1);
}
/*-----------------------------------------------------------------------
   is_crossing: tests a point on a contour to figure out if
   we are crossing the contour or tangent to the contour.  If we are
   tangent to it we don't want to bump the elevation yet.
   The test is performed by examining the six pixels to the side:
      1       4
           2 h,v 5
           3      6
   Crossings are:
                        two or more hits in range 1-6
                        1+ in 1-3 and 1+ in 4-6
   return false for a tangent, true for a crossing.
  ----------------------------------------------------------------------*/
static int is_crossing(h, v)
int       h, v;
{
        register int h_test, v_test, i;

        h_test = h - 1;       /* test the 1-3 */
```

(continued on next page)

GenView

3 D GRAPHICS ANIMATION

Now you can have a 3 D animating graphics system for your EGA or CGA equipped PC or PC compatible. (DOS 2.1 or higher.)

1 Includes all source code (C and ASM)
2 Hidden surface removal, translation, scaling and rotation
3 Scripting for viewpoint and object movements
4 Slideshow or animated playback of display frames
5 Software interface to single frame camera
6 "Fly through" of scenes with moving objects
7 Variable order spline curve generation for object and view point path generation
8 Object description includes MACRO capability
9 Source code includes direct to hardware assembly routines for high speed display
10 Control viewpoint path, acceleration, and velocity
11 Control object path, velocity, and rotation

GenView allows you to specify a viewpoint path with a small set of "turning points." It will then generate a smoothed series of frame locations along the path according to the velocity and acceleration parameters you specify. View direction may be backwards, forwards, or user specified. Object movement is controlled through a similar process with the addition of rotation (about any or all axes).

CGA Animation Sequence Demo
$7.00
EGA Animation Sequence Demo
$9.00 (2 diskettes)
CGA Flythrough Sequence Demo
$9.00 (Requires hard disk)

GenView System $99.00

The GenView System includes executable code, source, MAKE files, object description library, and manual (more than 40 pages) on diskette.

No Purchase Orders. Massachusetts residents add 5% sales tax.
Outside USA add $15.00.
VISA and MASTERCARD accepted.

Call 617-528-4280 to order or send check or money order to:

GenSoft Systems
Dept. 9D
P.O. Box 1
Foxborough, MA 02035

CIRCLE 166 ON READER SERVICE CARD

Listing Nine *(Listing continued, text begins on page 18.)*

```
        v_test = v - 1;
        for (i = 0; !getpixel(h_test, v_test, &bmap) && i < 3; i++, v_test++)
                ;

        if (i == 3)
                return false;           /* a tangent was hit */

        h_test = h + 1;         /* test the 4-6 */
        v_test = v - 1;
        for (i = 0; !getpixel(h_test, v_test, &bmap) && i < 3; i++, v_test++)
                ;

        if (i == 3)
                return false;           /* a tangent was hit */

        return true;                    /* looks good! */
}

/*-------------------------------------------------------------------------
   start_inflection: tests to see if this point is the start of an
   inflection in the contour. An inflection will not look like a crossing,
   but should be counted as one.
   An inflection looks like

          1
                h,v
                 2
                 3
                 4
                  5
        for example.  Only the first point on the inflection is counted.
 -------------------------------------------------------------------------*/
static int start_inflection(h, v)
int     h, v;
{
        register int h_test, v_test, dir = 0, i;

        if (getpixel(h, v-1, &bmap))
                return false;                                   /* only count the start of an inflection */

        h_test = h - 1;         /* test the 1-3 */
        v_test = v - 1;
        for (i = 0; !getpixel(h_test, v_test, &bmap) && i < 3; i++, v_test++)
                ;

        if (i < 3)
                dir = 1;

        h_test = h + 1;         /* test the 4-6 */
        v_test = v - 1;
        for (i = 0; !getpixel(h_test, v_test, &bmap) && i < 3; i++, v_test++)
                ;

        if (i < 3)
                dir = -1;

        if (dir == 0)
                return false;                   /* no inflection here */

        /*
                final test: track pixels downward
                                        if the turn at the end is in the opposite direction
                                        as the original turn (indicated by dir)
                                        then it is an inflection
        */

        v_test = v;
        while (getpixel(h, v_test+1, &bmap))
                v_test++;

        if (v_test != v) {
                v = v_test;

                /* let's test the pixels again! */
                h_test = h - 1;         /* test the 1-3 */
                v_test = v - 1;
                for (i = 0; !getpixel(h_test, v_test, &bmap) && i < 3; i++, v_test++)
                        ;

                if (i < 3)
                        if (dir != 1)
                                return true;            /* no change in direction, inflection */
                        else
                                return false;           /* reversed direction, was a tangent */

                h_test = h + 1;         /* test the 4-6 */
                v_test = v - 1;
                for (i = 0; !getpixel(h_test, v_test, &bmap) && i < 3; i++, v_test++)
                        ;

                if (i < 3)
                        if (dir != -1)
                                return true;            /* no change in direction, inflection */
                        else
                                return false;
        }
        else
                return false;
}

#ifdef DEBUG

/*
        Here is some code to display the progress of the algorithm
        Fixed point math is used to speed up calculations.
        Fixed point math is tricky in a typed language like C or
        Pascal, and a cinch in assembly.
*/

#include <QuickDraw.h>

static Fixed int_to_fix(i)
int i;
```

```
{
          asm {
                    move.w    i,d0      ; get the number
                    ext.l     d0                  ; clear top of d0
                    swap      d0                  ; make it a fixed point number, all done
          }
}

static int fix_to_int(i)
Fixed i;
{
          asm {
                    move.l    i,d0                ; get the number
                    add.l     #0xA000,d0; add .5 to number, for rounding
                    swap      d0                            ; make the int, ignore upper word of d0
          }
}

static Fixed ratio = 0x00004000;

static show_point(h,v)
register int h, v;
{
          /*
                    draw each point in right window
                    scaled like the MacPaint draw function
          */
          GrafPtr oldPort;
          Rect r;

          GetPort(&oldPort);
          SetPort(right_wind);

          r.top    = fix_to_int(FixMul(int_to_fix(v), ratio));
          r.left   = fix_to_int(FixMul(int_to_fix(h), ratio));
          r.right  = r.left + 1;
          r.bottom = r.top + 1;

          FrameRect(&r);
          SetPort(oldPort);
}

static show_line(h1,v1,h2,v2)
register int h1, v1, h2, v2;
{
          /*
                    draw each line in right window
                    scaled like the MacPaint draw function
          */
          GrafPtr oldPort;

          GetPort(&oldPort);
          SetPort(right_wind);

          MoveTo(fix_to_int(FixMul(int_to_fix(h1), ratio)),
                    fix_to_int(FixMul(int_to_fix(v1), ratio)));
          LineTo(fix_to_int(FixMul(int_to_fix(h2), ratio)),
                    fix_to_int(FixMul(int_to_fix(v2), ratio)));

          SetPort(oldPort);
}
#endif
```

End Listing Nine

Listing Ten

```
/*
          getpixel.h

          contains function prototype for getpixel.c
*/

#include <QuickDraw.h>

extern int getpixel(int h, int v, BitMap *bmap);
```

End Listing Ten

Listing Eleven

```
/*------------------------------------------------------------------------
          getpixel.c
          Returns the value of pixel h, v in the bitmap "bmap".
          Written in 68000 inline assembly both because it is faster
          and because it is quite easy to do.  A version of this code
          was used by Mike Morton in his StarFlight program.

          This version has the following limitations:

          1.  It assumes that the cursor will not be in the way,
          i.e. it has either been hidden or we are looking at an off-screen
          bitmap.  In the contour map algorithm we are looking at an
          off-screen bitmap.

          2.  This version also assumes that the coordinates at the upper left
          are (0,0). Any offset tc the coordinate system (by a call to SetOrigin)
          is ignored.

          3.  This code will probably not work on the screen buffer for large screen
          Macs, color Macs, gray scale Macs, etc.  It works fine on off-screen bitmaps
          on my Mac II, as long as the depth is 1 bit per pixel.

          William May
          303A Ridgefield Circle
          Clinton, MA 01510

          Created:  1/20/87
          Modified: 3/21/87   Found a bug: getpixel wouldn't handle
                              bitmaps > 32K.
  ----------------------------------------------------------------------*/
```

(continued on next page)

CQL

SQL Compatible Query System adaptable to any operating environment.

CQL Query System. A subset of the Structured English Query Language (SEQUEL, or SQL) developed by IBM. Linked files, stored views, and nested queries result in a complete query capability. File system interaction isolated in an interface module. Extensive documentation guides user development of interfaces to other record oriented file handlers.

Portable Application Support System

Portable Windowing System. Hardware independent windowing system with borders, attributes, horizontal and vertical scrolling. User can construct interface file for any hardware. Interfaces provided for PC/XT/AT (screen memory interface and BIOS only interface), MS-DOS generic (using ANSI.SYS), Xenix (both with and without using the curses interface), and C-library (no attributes).

Screen I/O, Report, and Form Generation Systems. Field level interface between application programs, the Query System, and the file system. Complete input/output formatting and control, automatic scrolling on screens and automatic pagination on forms, process intervention points. Seven field types: 8-bit unsigned binary, 16 bit signed binary, 16 bit unsigned binary, 32 bit signed binary, monetary (based on 32 bit binary), string, and date.

Including Source Code

$395.00

File System interfaces include C-tree and BTRIEVE.

HARDWARE AND FILE SYSTEM INDEPENDENT

MACHINE INDEPENDENT SOFTWARE CORPORATION

1415 NORTHGATE SQUARE #21D
RESTON, VA 22090

VISA/Master Charge accepted
(703) 435-0413

*C-tree is a trademark of FairCom

IBM, SEQUEL, PC, XT, AT are trademarks of IBM Corp.
MS-DOS and Xenix are trademarks of Microsoft Corp.

CQL and the CQL logo are trademarks of Kurtzberg Computer Systems.

CIRCLE 294 ON READER SERVICE CARD

Listing Eleven *(Listing continued, text begins on page 18.)*

```
#include <QuickDraw.h>
#include <asm.h>

#include "getpixel.h"

int getpixel(h, v, bmap)
int h, v;
BitMap *bmap;
{
        asm {
                move.w    v,d0                                 ; load v and h into registers
                ext.l     d0
                move.w    h,d1
                ext.l     d1
                move.w    d1,d2                                ; copy h coord for bit offset
                                                               ; only need low order byte

                move.l    bmap,a0                              ; point to bitmap

                mulu      OFFSET(BitMap,rowBytes)(a0),d0
                                                               ; v * rowbytes is offset to row
                lsr.l     #3,d1                                ; extract byte offset
                add.l     d1,d0
                not.b     d2                                   ; make bit number 68000 style
                move.l    OFFSET(BitMap,baseAddr)(a0),a0
                                                               ; get base address of bitmap
                add.l     d0,a0                                ; get to the correct byte
                btst      d2,(a0)                              ; test the bit
                beq                @0

                moveq     #1,d0                                ; set value to 1
                return
@0
                moveq     #0,d0                                ; set value to 0
        }
}
```

End Listing Eleven

Listing Twelve

```
/*
        mem.h

        definition info for memory management

        William May

        created:  3/25/87
*/

#define SYSERR -1

/*
 * roundew, truncew - round up or truncate address to the next
 * even word.
 */

#define roundew(x)    (int *)((3 + (long)(x)) & (~3))
#define truncew(x)    (int *)(((long)(x)) & (~3))

#define DEFSIZE  (150000)       /* default size for a new pool */

/*
 * node structure for each node in the free memory list
 */
struct mblock {
        struct mblock *mnext;
        unsigned long mlen;
};

/*
 * structure for pools
 */
struct pool {
        struct pool *pnext;
        struct mblock firstblock;
};
```

End Listing Twelve

Listing Thirteen

```
/*-----------------------------------------------------------------
        getmem.c

        Implements low overhead memory management for nonrelocatable
        blocks on the Macintosh.  This code is based on "Operating System
        Design: The XINU Approach" by Douglas Comer (Prentice Hall, 1984.
        Great book!)

        Motivation:

        The Macintosh Memory Manager is designed to use relocatable
        blocks (i.e. pointers to pointers) for heap data structures.  In
        order that relocatable blocks be free to move the Memory Manager
        tries to keep nonrelocatable blocks (i.e. blocks referred to by
        pointers) out of the way, i.e. in low mem.  To do this, when a program
        asks for a nonrelocatable block the Memory Manager starts a linear
        search for available space at the bottom of the heap.  In an
        application such as a tree, where many thousands of small blocks are
        needed, the linear search technique quickly dies.

        The code below bypasses this problem by requesting one large block at
        the start of execution (i.e. when init_mem is called).  The function
        getmem then maintains a pointer to the next available block.  Thus
        any request for memory is satisfied immediately.  No search is needed.
        The performance difference in the contour analysis is quite dramatic.
```

Cross Assemblers
Universal Linker
Powerful Librarian

PC/MS DOS, micro VAX,
VAX VMS, VAX UNIX/ULTRIX

- Targeting over 30 Microprocessors
- Version 2.1 is FAST
- Powerful Macros
- Absolute or Relocatable Code
- Compatible with all Assemblers
- Conditional Assembly
- $295 up for Complete Packages

Enertec inc.
19 Jenkins Ave.
Lansdale, PA 19446 U.S.A.
telephone: 215-362-0966
telex: 4948709 ENERTEC

CIRCLE 346 ON READER SERVICE CARD

PLOT TEXT ON ANY GRAPHICS SCREEN!!

YES, we said ANY Graphics Screen, even VGA! FINALLY! Xgraf is a super set of smart low level assembly graphic routines that you call directly from Compiled BASIC. Xgraf replaces BASIC's confusing graphics statements with consistent, full featured calls specifically designed for the BASIC programmer.

FINALLY! Xgraf is only $99.00 + $4.00 S&H

We specialize in libraries and tools for Compiled BASIC. Our catalog features the FINALLY! Family of Products and other top flight tools.

Call: 1 800 423-3400
(9:00 AM to 8:00 PM EST)
PA & AK call (412) 782-0384

KOMPUTERWERK
851 Parkview Blvd.
Pittsburgh, PA 15215

CIRCLE 388 ON READER SERVICE CARD

***NEW!* TLIB™ 4.0**
SOURCE CODE CONTROL
The best keeps getting better!

The critics *loved* TLIB 3.0...
"...packed with features... [generates deltas] amazingly fast... [of the 6 reviewed] the two best packages are Burton Systems' TLIB and [a $395 product], so designated because of their ease of use, abundance of features, and ability to be configured..." ***PC Tech Journal Sept 87***
"...has my highest recommendation." *Ronny Richardson,* ***Computer Shopper Aug 87***

- The **fastest, most powerful** system is now even faster!
- ***Many new features!*** Expanded keyword support. Multi-line comments. Branching, for multiple development lines. Extended wildcard and list-of-file support; creates lists by scanning source code for includes. Can merge (reconcile) multiple simultaneous changes and undo intermediate revisions. Network and WORM optical disk support.
- Includes a copy of Landon Dyer's excellent public domain **MAKE** utility (with source code for DOS & VAX/VMS).

PC/MS-DOS 2.x & 3.x ***Just $99.95*** *+ $3 s/h* Visa/MC
BURTON SYSTEMS SOFTWARE
P. O. Box 4156, Cary, NC 27519-4156
(919) 469-3068

CIRCLE 212 ON READER SERVICE CARD

Listing Thirteen *(Listing continued, text begins on page 18.)*

```
          The code below is quite simple but not flexible.  One notable
          lack is for a routine to free memory.  Such a function is shown in
          Comer's book.  Another limitation that only one block is used.
          The functions can be extended to be able to add new blocks when it
          needs additional space.


          William May
          303A Ridgefield Circle
          Clinton, MA 01510

          created:  3/25/87     very primitive version, but works:
                                one pool only created
                                user has to guess the required space
                                speed increase is dramatic, especially
                                as the contour algorithm progresses (i.e.
                                as the AVL tree gets quite large).
  ------------------------------------------------------------------- */

#include <MemoryMgr.h>
#include "mem.h"

struct mblock memlist;                  /* head of free memory list */
struct pool *plist;                     /* head of pool list */

/*------------------------------------------------------------------
          init_mem creates a large memory pool, and initializes the necessary
          data_structures.
  -------------------------------------------------------------------*/
int init_mem()
{
          plist = (struct pool *)NewPtr(sizeof(struct pool) + DEFSIZE);

          if (MemErr == noErr) {
                    plist->pnext = (struct pool *)0;

                    memlist.mnext  = &(plist->firstblock);
                    (memlist.mnext)->mnext = (struct mblock *)0;
                    (memlist.mnext)->mlen  = (long)(DEFSIZE);
          }

          return MemErr;
}

/*------------------------------------------------------------------
          getmem allocates memory in the pool.  On successful completion
          a pointer to the allocated space is returned to the caller.  Otherwise
          a null pointer (0L) is returned.
  -------------------------------------------------------------------*/
int *getmem(nbytes)
unsigned long nbytes;
{
           register struct mblock *p, *q, *leftover;

           if (nbytes == 0) {
                     return (int *)0;
           }

           nbytes = (unsigned long)roundew(nbytes);

           for (q=&memlist, p=memlist.mnext; p != 0L; q=p, p=p->mnext)
                     if (p->mlen == nbytes) {
                               q->mnext = p->mnext;
                               return ((int *)p);
                     } else if (p->mlen > nbytes) {
                               leftover = (struct mblock *)((unsigned long)p + nbytes);
                               q->mnext = leftover;
                               leftover->mnext = p->mnext;
                               leftover->mlen = (long)(p->mlen - nbytes);
                               return ((int *)p);
                     }

            return ((int *)0);

}
```

End Listing Thirteen

Listing Fourteen

```
 /*------------------------------------------------------------
           error.c

           very primitive error handler

           William May
           303A Ridgefield Circle
           Clinton, MA 01510
   ------------------------------------------------------------*/

 #include <stdio.h>

 /*------------------------------------------------------------
           if an error number specified then print it,
           otherwise only print the message
   ------------------------------------------------------------*/
 syserr(errno, s)
 int errno;
 char *s;
 {
           if (errno) printf("Error (%d): %s\n", errno, s);
           else printf("Error: %s\n", s);

           ExitToShell();
 }
```

End Listings

ZyINDEX
Your personal researcher™

FULL TEXT RETRIEVAL

NEW!!
$95 PERSONAL VERSION

FIND

Searches 5,000 text files in 5 seconds.

The ultimate personal computer information retrieval software.
Find any information created by the popular word processors or ASCII files, in seconds-without having to set up a database or do programming. Scan a 20 Mbyte hard disk or a collection of floppies and instantly display all occurrences of a name, a product, a phrase, a number, or anything else you need to find. Save hours of manual searching.

TEXT

FEATURES
• **Comprehensive searches** create a search request with any combination of AND, OR, NOT, and WITHIN (WITHIN refers to how far apart two words or phrases can be — up to 30,000 words).
• **Wild Cards,** for example: type "micro*" to get (microcomputer, microcomputing, micro-processor, etc.). • **Mark and Save** retrieved information to create a new file.
• **Highlights** search-words and phrases in retrieved text. • **On Line Help Menus.**
• **Find Function** automatically displays search topic in the retrieved file. • **Phrase Search,** in addition to single word search.

SYSTEM REQUIREMENTS MINIMUM
• 384K • Two Disk Drives • DOS 2.0 or above • Designed for IBM PC, XT, AT, compatibles, and most MS-DOS computers

FAST

ZyINDEX Personal $95
Searches up to 325 files
ZyINDEX Standard $145
Searches up to 500 files
ZyINDEX Professional $295
Searches up to 5,000 files
ZyINDEX Plus $695
Searches up to 15,000 files with LAN capabilities.
30 Day Money Back Guarantee

ZyLAB™ Corporation
3105-T North Frontage Rd.
Arlington Heights, Illinois (312) 642-2201
(800) 544-6339 For orders and information.

CIRCLE 329 ON READER SERVICE CARD

8031

FORTH DEVELOPMENT ENVIRONMENT

Take advantage of Bryte's tools to make your job easier:

- Bryte's development environment uses BRYTE-FORTH on the actual production hardware during product development. No emulators, no changes, no surprises.
- Optional PC-based cross-development tools use DOS files as microcontroller mass storage. These files can be used to generate compact EPROM images, detailed listings, and cross-references.

Why not start developing the Bryte way today?

BRYTE-FORTH 8Ø31 EPROM (includes 13Ø page User's Manual)	100.00
Utility disk(s)	65.00*
Cross-compiler/Cross-assembler	235.00*
8Ø31 unlimited quan. license	1000.00*

* Includes complete source code

bryte computers, inc.

P.O. Box 46
Augusta, ME 04330-0046

207/547-3218

CIRCLE 387 ON READER SERVICE CARD

386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•

NOW!

386

C and Pascal

for MS-DOS

MetaWare Incorporated announces the *first* available C and Pascal compilers that generate

protected-mode 80386 code

for running on any 80386 machine that runs MS-DOS (e.g., the Compaq Deskpro 386). The compilers are functionally identical to the well-respected 8086/286 MS-DOS High C™ and Professional Pascal™ compilers that have received outstanding reviews in such magazines as Computer Language, Dr. Dobb's, and PC Tech Journal. Our compilers are currently used by industry leaders such as Ashton-Tate, AutoDesk, ANSA, and Lifetree. Now you can get them generating 80386 code.

If you have an application that requires the large 32-bit address space and the full 32-bit registers of the 80386, expand your marketplace to the rapidly growing supply of 80386 MS-DOS machines. Contact MetaWare for your 80386 software solution today! **(408) 429-6382, telex 493-0879.**

Durable Software Constructed Automatically™

Meta Ware™
INCORPORATED

903 Pacific Avenue, Suite 201 • Santa Cruz, CA 95060-4429

CIRCLE 95 ON READER SERVICE CARD

TURBO C TOOLBOX

Listing One *(Text begins on page 30.)*

```
/* VIDEO.I: Contains ROM BIOS video calls to be used in Turbo C */

#define  ROM  0x10

union REGS  inreg, outreg;

int videomode (int *ncols)                /* get current display mode */
{                                   /* return number of cols via *ncols */
  inreg.h.ah = 0x0F;
  int86 (ROM, &inreg, &outreg);
  *ncols = outreg.h.ah;                          /* number of cols */
  return (outreg.h.al);                             /* return mode */
} /* ------------------------- */
int activepage (void)                /* return active display page */
{
  inreg.h.ah = 0x0F;
  int86 (ROM, &inreg, &outreg);
  return (outreg.h.bh);
} /* ------------------------- */
void setmode (int mode)                          /* set video mode */
{
  inreg.h.al = mode;
  inreg.h.ah = 0x00;
  int86 (ROM, &inreg, &outreg);
} /* ------------------------- */
void setcursor (int start, int end)            /* set cursor shape */
{
  inreg.h.ch = start;
  inreg.h.cl = end;
  inreg.h.ah = 0x01;
  int86 (ROM, &inreg, &outreg);
} /* ------------------------- */
int curstart (void)                   /* get cursor starting line */
{
  inreg.h.bh = 0;              /* (cursor shape same in all pages */
  inreg.h.ah = 0x03;
  int86 (ROM, &inreg, &outreg);
  return (outreg.h.ch);
} /* ------------------------- */
int cursend (void)                      /* get cursor ending line */
{
  inreg.h.bh = 0;
  inreg.h.ah = 0x03;
  int86 (ROM, &inreg, &outreg);
  return (outreg.h.cl);
} /* ------------------------- */
void cursoff (void)                             /* turn cursor off */
{
  inreg.h.ch = curstart () | 0x10;               /* turn on bit 4 */
  inreg.h.cl = cursend ();
  inreg.h.ah =  0x01;
  int86 (ROM, &inreg, &outreg);
} /* ------------------------- */
void curson (void)                               /* turn cursor on */
{
  inreg.h.ch = curstart () & 0x07;     /* turn off high order bits */
  inreg.h.cl = cursend ();
  inreg.h.ah =  0x01;
  int86 (ROM, &inreg, &outreg);
} /* ------------------------- */
void gotoxy (int col, int row, int page)         /* set cursor pos */
{                                           /* must specify page */
  inreg.h.bh = page;
  inreg.h.dh = row;
  inreg.h.dl = col;
  inreg.h.ah = 0x02;
  int86 (ROM, &inreg, &outreg);
} /* ------------------------- */
int wherex (int page)              /* return cursor column in page */
{
  inreg.h.bh = page;
  inreg.h.ah = 0x03;
  int86 (ROM, &inreg, &outreg);
  return (outreg.h.dl);
} /* ------------------------- */
int wherey (int page)                 /* return cursor row in page */
{
  inreg.h.bh = page;
  inreg.h.ah = 0x03;
  int86 (ROM, &inreg, &outreg);
  return (outreg.h.dh);
} /* ------------------------- */
void setpage (int page)               /* set active display page */
{
  inreg.h.al = page;
  inreg.h.ah = 0x05;
  int86 (ROM, &inreg, &outreg);
} /* ------------------------- */
void cls (void)                             /* clear active screen */
{
  inreg.h.al = 25;                               /* entire screen */
  inreg.h.bh = 0x07;                       /* set to gray on black */
  inreg.h.ah = 0x06;
  inreg.h.cl =  0; inreg.h.ch =  0;
  inreg.h.dl = 79; inreg.h.dh = 24;
  int86 (ROM, &inreg, &outreg);
  gotoxy (0, 0, activepage ());
} /* ------------------------- */
void window (int  x1, int y1,          /* window upper left corner */
             int  x2, int y2,                /* lower right corner */
             char attrib)              /* text attribute inside */
{
  inreg.h.al = y2 - y1 + 1;               /* clear entire window */
  inreg.h.bh = attrib;
  inreg.h.cl = x1; inreg.h.ch = y1;
  inreg.h.dl = x2; inreg.h.dh = y2;
  inreg.h.ah = 0x06;
  int86 (ROM, &inreg, &outreg);
} /* ------------------------ */
void winScroll (int x1, int y1, int x2, int y2,   /* scroll window */
                int attr)                         /* one line upward */
{
  inreg.h.al = 1;
  inreg.h.cl = x1; inreg.h.ch = y1;
  inreg.h.dl = x2; inreg.h.dh = y2;
  inreg.h.bh = attr;
  inreg.h.ah = 0x06;
  int86 (ROM, &inreg, &outreg);
} /* ------------------------ */
char chattr (int foregrnd, int backgrnd)          /* character attrib*/
{
  return ((backgrnd << 4) + foregrnd);
} /* ------------------------ */
char rdchara (int page,                        /* read char at curs pos */
              char *attrib)              /* return attribute indirectly */
{
  inreg.h.bh = page;
  inreg.h.ah = 0x08;
  int86 (ROM, &inreg, &outreg);
  *attrib = outreg.h.ah;
  return (outreg.h.al);
} /* ------------------------ */
void wrtcha (char ch, char attrib,              /* write char + attrib */
             int page)                         /* at cursor pos on page */
{                                        /* NOTE: does not advance cursor */
  inreg.h.al = ch;
  inreg.h.bh = page;
  inreg.h.bl = attrib;
  inreg.x.cx = 1;
  inreg.h.ah = 0x09;
  int86 (ROM, &inreg, &outreg);
} /* ------------------------ */
void wrtstra (char *str,                                /* write string */
              char attrib,                          /* with attribute */
              int page)                                   /* to page */
{                                      /* starting at cursor position */
int   c, r, n, p = 0;

  videomode (&n);                             /* get width of screen */
  r = wherey (page);                              /* get current row */
  while (str[p]) {
    wrtcha (str[p++], attrib, page);              /* write next char */
    if ((c = wherex (page)) < (n - 1))
      gotoxy (c + 1, r, page);                     /* advance cursor */
    else
      gotoxy (0, ++r, page);                           /* else wrap */
  }
} /* ------------------------ */
void wrtch (char ch, int color,                 /* write char in color */
            int page)                   /* at cursor position on page */
{
  inreg.h.al = ch;
  inreg.h.bl = color;
  inreg.h.bh = page;
  inreg.x.cx = 1;
  inreg.h.ah = 0x0A;
  int86 (ROM, &inreg, &outreg);
} /* ------------------------ */
void wrtstr  (char *str,                                /* write string */
              char color,                                   /* in color */
              int page)                                    /* to page */
{                                      /* starting at cursor position */
int   c, r, n, p = 0;

  videomode (&n);                             /* get width of screen */
  r = wherey (page);                              /* get current row */
  while (str[p]) {
    wrtcha (str[p++], color, page);               /* write next char */
    if ((c = wherex (page)) < (n - 1))
      gotoxy (c + 1, r, page);                     /* advance cursor */
    else
      gotoxy (0, ++r, page);                           /* else wrap */
  }
} /* ------------------------ */
void palette (int palno)                          /* set color palette */
{                                      /* valid only in mode 4 on CGA */
  inreg.h.bh = 1;
  inreg.h.bl = palno;
  inreg.h.ah = 0x0B;
  int86 (ROM, &inreg, &outreg);
} /* ------------------------ */
void graphbackground (int color)           /* set graphics b/g color */
{
  inreg.h.bh = 0;
  inreg.h.bl = color;
  inreg.h.ah = 0x0B;
  int86 (ROM, &inreg, &outreg);
} /* ------------------------ */
void plot (int x, int y, int pixel)              /* plot pixel at x, y */
{                                               /* pixel (color) value */
  inreg.h.al = pixel;
  inreg.h.bh = 0;
  inreg.x.cx = x;
  inreg.x.dx = y;
  inreg.h.ah = 0x0C;
  int86 (ROM, &inreg, &outreg);
} /* ------------------------ */
int pixel (int x, int y)                 /* return pixel value at x, y */
{
  inreg.x.cx = x;
  inreg.x.dx = y;
  inreg.h.ah = 0x0D;
  int86 (ROM, &inreg, &outreg);
  return (outreg.h.al);
} /* ------------------------ */
```

End Listing One

(Listing Two begins on next page)

Announcing WKS LIBRARY **NEW!**

The Lotus "Wrap-Around" for C Programs

Now you can write <u>and read</u> Lotus worksheets directly from a C program!

WKS LIBRARY lets you:

- avoid time-consuming file translation steps
- control the execution of 1-2-3 from inside your application
- use Lotus as a data entry screen in your C program
- generate "live" financial statements with formulas for totals

Feature this:

- reads and writes .WKS, .WK1 and .WR1 worksheets
- writes integers, floats, strings, formulas, macros and dates using *wprintf()*
- reads using *wscanf()*, converting column contents to C variables according to format specs
- has low-level functions for manipulating individual cells
- provides Lotus control functions such as: formats, column widths, initial cursor position, range names, cell protection
- supports Lattice, Microsoft & Turbo compilers for DOS, plus most UNIX environments

Source Code provided **No Royalties on executable programs**

Only $89 (includes free 800-line support)

ORDER TODAY! **Toll-free (800) 367-9882**

Tenon Software, Inc.
1980 - 112th NE, #250, Bellevue, WA 98004
(206) 453-1914 (in Washington state)

Amazing COMPUTING™

Your Original AMIGA™ Monthly Resource

<u>*FEATURING*</u>

- •Complete Amiga Hardware and Software reviews
- •A vast and growing library of over 110 PDS Disks
- •Solid and informative for both the advanced and beginning Amiga User
- •Understandable program listings and tools
- •Step by Step Hardware projects

Amiga Users have made Amazing Computing™ the longest running Monthly magazine dedicated to the Commodore Amiga. If you are searching for Amiga technical information that is both current and comprehensive, then be amazed by the pioneer Amiga Magazine, Amazing Computing - your Original AMIGA Monthly Resource.

YES, Amaze Me! I have enclosed $24.00 U.S. ($30.00 Canada & Mexico, $35.00 Overseas) in check or money order (U.S. funds drawn on a U.S. bank) to:

PiM Publications, Inc.
P.O. Box 869-DD
Fall River, MA 02722

Name ____________________
Address ____________________
City ____________ St ______ Zip ______

CIRCLE 136 ON READER SERVICE CARD

Listing Two *(Text begins on page 30.)*

```
/* VIDEO.H: Prototypes for contents of user-written VIDEO.LIB */
/* Describes calls to ROM BIOS video functions */

int videomode (int *ncols);
int activepage (void);
void setmode (int mode);
void setcursor (int start, int end);
int curstart (void);
int cursend (void);
void cursoff (void);
void curson (void);
void gotoxy (int col, int row, int page);
int wherex (int page);
int wherey (int page);
void setpage (int page);
void cls (void);
void window (int x1, int y1, int x2, int y2, char attrib);
char chattr (int fgrnd, int bgrnd);
char rdchara (int page, char *attr);
void wrtcha (char ch, char attr, int page);
void wrtstra (char *str, char attr, int page);
void wrtch (char ch, int color, int page);
void wrtstr (char *str, char color, int page);
void palette (int palno);
void graphbackground (int color);
void plot (int x, int y, int pixel);
int pixel (int x, int y);
```

End Listing Two

Listing Three

```
/* DRAW.I: Draws lines in graphics mode */

/* hdraw draws horizontal line along y between x1 and x2 */
void hdraw (int x1, int x2, int y, int color)
{
int    x;

  if (x1 > x2) {                /* sort x's into left-to-right order */
    x = x1; x1 = x2; x2 = x;
  }
  for (x = x1; x <= x2; x++)
    plot (x, y, color);
} /* ------------------------ */

/* vdraw draws vertical line along x between y1 and y2 */
void vdraw (int y1, int y2, int x, int color)
{
int    y;

  if (y1 > y2) {                 /* sort y's into top-to-bottom order */
    y = y1; y1 = y2; y2 = y;
  }
  for (y = y1; y <= y2; y++)
    plot (x, y, color);
} /* ------------------------ */

/* draw() does lines on the diagonal */
void draw (int x1, int y1, int x2, int y2, int color)
{
double   xstep, ystep, xcum = 0.0, ycum = 0.0;
int      dx, dy;                                        /* deltas */
register x, y;

  dx = x2 - x1;
  dy = y2 - y1;
  if (abs (dx) >= abs (dy)) {                   /* plot along x axis */
    ystep = (double) dy / dx;        /* movement along y axis per x */
    if (dy < 0) {                           /* y travels to the left */
      if (ystep > 0)
        ystep *= -1;              /* adjust for wrong sign from -y/-x */
    } else                                 /* y travels to the right */
      if (ystep < 0)
        ystep *= -1;                             /* adjust as above */
    dx /= abs (dx);                                  /* x increment */
    for (x = x1, y = y1; x != x2; x += dx) {
      plot (x, y, color);
      ycum += ystep;                       /* cum motion along y axis */
      y = y1 + ycum;                                      /* next y */
    }
  } else {                                      /* plot along y axis */
    xstep = (double) dx / dy;        /* movement along x axis per y */
    if (dx < 0) {
      if (xstep > 0)
        xstep *= -1;
    } else
      if (xstep < 0)
        xstep *= -1;
    dy /= abs (dy);                                  /* y increment */
    for (y = y1, x = x1; y != y2; y += dy) {
      plot (x, y, color);
      xcum += xstep;                       /* cum motion along x axis */
      x = x1 + xcum;                                      /* next x */
    }
  }
} /* -------------------------- */
```

End Listing Three

Get Real.

Get real productive with REAL-TOOLS, a general purpose set of "C" development tools for UNIX™ and XENIX™.

Get Graphics Too! In addition to an advanced screen management system and superior windowing capabilities, REAL-TOOLS offers user-defined graphics for you to draw, save, recall, copy and animate symbols and panels.

So if you're developing applications for the real world — get real productive. Get graphics. Get REAL-TOOLS.

Real-Tools™

$99 Binary only. $549 Library source. $999 Complete source.

PCT

Pioneering Controls Technologies, Inc.
510 Bering Drive, Suite 300, Houston, Texas 77057
(713) 266-8649

™REAL-TOOLS is a trademark of Pioneering Controls Technologies, Inc.
™UNIX is a trademark of AT&T
™XENIX is a trademark of Microsoft Corporation

CIRCLE 191 ON READER SERVICE CARD

NEW!
FASTER THAN EVER!
DeSmet C V3.0

FASTER C DEVELOPMENT

Invoke the DeSmet C compiler from the SEE™ full screen editor and the first error will return you immediately to SEE at the error line with the error message displayed.

FASTER COMPILATION

When you don't use inline assembly code or don't want to see the ASM88 output, the V3.0 compiler produces object code directly - making DeSmet C up to ***twice as fast*** as before.

PLUS EXPANDED STANDARD LIBRARY

Networking, path, file, time, enhanced string functions, environment support now included.

FULL FEATURES WITH EVERY PACKAGE
— ONLY $109 —

C Compiler, Assembler, Binder, Librarian, Execution Profiler, Overlays, 8087 and S/W Floating Point, Full STDIO Library and Full Screen Editor (SEE). Debugger and Large Case options available at $50 each.

C Ware Corporation
P.O. Box 428, Paso Robles, CA 93447
Phone: (805) 239-4620 **Telex:** 358185 **BBS:** (805) 239-4834
We accept **VISA, MC & AMEX.** Call now and we'll ship today.
Street Address: 945 Spring #14, Paso Robles, CA 93446

Listing Four

```
/* COLORS.H: Maps color names */

#define BLACK          0
#define BLUE           1
#define GREEN          2
#define CYAN           3
#define RED            4
#define MAGENTA        5
#define BROWN          6
#define LIGHTGRAY      7
#define DARKGRAY       8
#define LIGHTBLUE      9
#define LIGHTGREEN     10
#define LIGHTCYAN      11
#define LIGHTRED       12
#define LIGHTMAGENTA 13
#define YELLOW         14
#define WHITE          15
```

End Listing Four

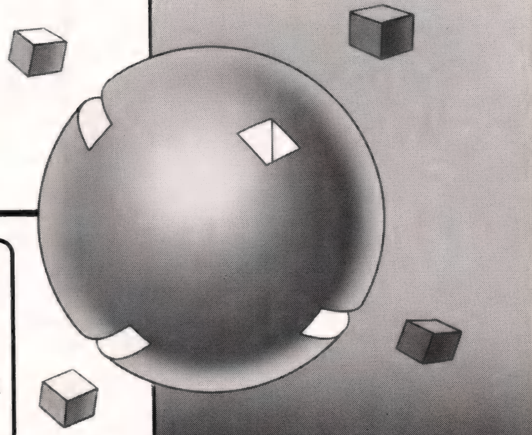
Listing Five

```
/* VID.C: Demos video functions */

/* TURBO C INCLUDES */
#include <stdio.h>
#include <bios.h>
#include <dos.h>

/* USER-WRITTEN INCLUDES */
#include <colors.h>
#include <video.i>
#include <draw.i>

/* LOCAL FUNCTION PROTOTYPES */
int  vidIdent (int *vidmode);
void wait (void);
void stairsteps (void);
int  isEGA (void);
void label (void);
void bigX (int adap, int vmode);
void hourglass (int adap, int vmode);

/* GLOBALS */
enum vidTypes {mda, cga, ega, compaq, other};
```

```
main ()
{
int  adaptor, mode, cols;

  cls ();                                                /* clear screen */
  adaptor = vidIdent (&mode);                   /* identify video adaptor */
  if (adaptor != other) {
    stairsteps ();                             /* cursor positioning demo */
    bigX (adaptor, mode);                             /* graphics demo #1 */
    hourglass (adaptor, mode);                        /* graphics demo #2 */
    label ();
    gotoxy (36, 12, 0);
    puts ("All done!");
  }
} /* ------------------------ */
int  vidIdent (int *vidmode)                    /* identify video adaptor */
{
int  flag, adap, width;

  label ();                                             /* label display */
  puts ("\n\nDISPLAY INFORMATION:");
  *vidmode = videomode (&width);                       /* get video mode */
  flag = (biosequip () & 0x18) >> 4;             /* get video eqpt flag */
  if (isEGA ()) {
    adap = ega;
    puts ("\n\n  Enhanced Graphics Adaptor");
  } else
    switch (flag) {
      case 0: if (*vidmode == 2) {
                adap = compaq;
                puts ("\n\n  Compaq adaptor");
              } else {
                adap = cga;
                puts ("\n\n  Color Graphics Adaptor");
              }
              break;
      case 3: adap = mda;
              puts ("\n\n  Monochrome Display Adaptor");
              break;
      default: adap = other;
              puts ("\n\n  Adaptor not usable in this demo. Sorry.");
    } /* end of switch */
  printf ("\n  Text screen size is %d columns x 25 rows\n", width);
  if ((*vidmode < 4) || (*vidmode == 7))
    puts ("\n  Text mode currently active");
  else
    puts ("\n  Graphics mode currently active");
  wait ();
  return (adap);
```

(continued on next page)

C PROGRAMMERS! THE TOOLS YOU NEED AT A PRICE YOU'LL LIKE



BTree Supports all index file operations. Very quick sequential or random access, duplicate keys, multiple indices, fixed and variable length data records are all supported. **75.00**

ISAM Works on top of BTree to provide a simple, yet powerful application program/file system interface. Complex filesystem manipulation becomes a snap. Provides the power of a database manager with the flexibility of a programming language. **40.00**

NOW MULTI -USER AVAILABLE 60.00

lp Finally, a completely device independent printer library! lp drives any printer as accurately as possible and allows easy access to its most sophisticated features. Multiple fonts, multi-column output, complex margin formatting, and much more. Pays for itself the first time it's used. **75.00**

Snake The ultimate 'make' utility. We couldn't find a good one, so we wrote a great one. Has all kinds of powerful features including wild card filename expansion, nested macros, and multiple dependency and rules definitions. Ready to go for MS-DOS; C source is there if you use another operating system. **59.00**

Combine & Save: BTree + ISAM + lp **159.00** + Snake **199.00**

Each product includes a typeset manual, example programs, and complete C source code that runs on any operating system. Softfocus products may be incorporated into applications royalty-free.

Credit card orders accepted. Visa, M/C, Amex. Dealer inquiries invited.

softfocus
1343 Stanbury Drive
Oakville, Ontario, Canada
L6L 2J5
(416) 825-0903

CIRCLE 259 ON READER SERVICE CARD

New!

386|DEBUG

- A symbolic debugger for 80386 32-bit *protected mode* programs which run under Phar Lap's 386|DOS-Extender™
- Breakpoints, data watchpoints, and built-in disassembler
- Fully compatible with Phar Lap's **386|ASM/LINK**, the MetaWare 80386 **High C™** and **Professional Pascal™** compilers, and the Green Hills 80386 **Fortran** compiler
- Runs on all DOS-based PCs equipped with an 80386 CPU, including the Compaq® DESKPRO 386™, the IBM®PS/2™ Model 80, and most accelerator cards, including the Intel Inboard™ 386/AT
- $195—Available *today*

(617) 661-1510

Phar Lap Software, Inc.
60 Aberdeen Ave.
Cambridge, MA 02138

"The 80386 Software Experts"

CIRCLE 343 ON READER SERVICE CARD

Now with dBx EXPRESS

dBASE to C

the dBx™ Translator

- C from dBASE II, III, III+ programs
- Move to UNIX, XENIX, QNX, MAC, AMIGA
- Faster, more reliable IBM PC programs
- Supports multi-user and network
- Run your code on any standard C system
- Know dBASE? Learn C easily.
- Priced from $350; available from distributors
- Includes full screen handler library
- Supports a choice of C database managers

from Desktop Ai

1720 Post Road East, Westport, CT 06880
Telephone: 203-255-3400 • Telex: 6502972226MCI
MCIMAIL-DESKTOPAI

dBASE is a trademark of Ashton-Tate   dBx is a trademark of Desktop AI

CIRCLE 258 ON READER SERVICE CARD

Listing Five

(Listing continued, text begins on page 30.)

```
} /* ------------------------- */
int  isEGA (void)                  /* determine if EGA is attached */
{                                /* return TRUE if so, FALSE if not */
  return (peekb (0x40, 0x87));              /* check EGA info byte */
} /* ------------------------- */

void wait (void)           /* prompt to continue, wait for keypress */
{
int   tab, width;

  videomode (&width);                       /* get width in columns */
  tab = (width - 33) / 2;              /* starting column for text */
  gotoxy (tab, 24, 0);
  wrtstr ("Press any key to continue demo...", WHITE, 0);
  getch ();
  cls ();
} /* ------------------------- */
void label (void)                   /* label the screen at top center */
{
  gotoxy (30, 0, 0);
  wrtstra ("Video demonstration", chattr (YELLOW, BLUE), 0);
} /* ------------------------- */
void stairsteps (void)
{
int   color = 1;

   label ();
   gotoxy (31,  2, 0); wrtstr ("Cursor positioning", LIGHTGRAY, 0);
   gotoxy (10,  4, 0); wrtstr ("Stair", color++, 0);
   gotoxy (20, 10, 0); wrtstr ("steps", color++, 0);
   gotoxy (30, 16, 0); wrtstr ("going", color++, 0);
   gotoxy (40, 22, 0); wrtstr ("down",  color++, 0);
   gotoxy (50, 16, 0); wrtstr ("and",   color++, 0);
   gotoxy (60, 10, 0); wrtstr ("back",  color++, 0);
   gotoxy (70,  4, 0); wrtstr ("up",    color, 0);
   wait ();
} /* ------------------------- */
void bigX (int vidAdap, int vmode)           /* APA graphics demo #1 */
{                    /* draws full-screen border and X, adjusting */
                                          /* for EGA or CGA as indicated */
int  x1 =  0, x2 = 639;
int  y1 = 15, y2;

  if ((vidAdap == mda) || (vidAdap == other))        /* can't do it */
    return;                              /* so return with no action */

  if (vidAdap == ega) {             /* EGA demo: 640 x 350 (mode 0Fh) */
    y2 = 349 - 15;                   /* set bottom of graphics screen */
    setmode (0x0F);                 /* go to EGA mono graphics mode */
  } else {                /* CGA|Compaq demo: 640 x 200 (mode 06h) */
    y2 = 199 - 15;
    setmode (0x06);
  }
  label ();                                     /* label the screen */
  hdraw (x1, x2, y1, 1);                     /* draw line across top */
  hdraw (x1, x2, y2, 1);                              /* then bottom */
  vdraw (y1, y2, x1, 1);                           /* down left side */
  vdraw (y1, y2, x2, 1);                          /* down right side */
  draw (x1, y1, x2, y2, 1);                          /* main diagonal */
  draw (x1, y2, x2, y1, 1);                         /* cross diagonal */
  wait ();
  setmode (vmode);
} /* ------------------------- */
void hourglass (int vidAdap, int vmode)          /* graphics demo #2 */
{                   /* operates in 320 x 200 four-color (CGA) mode */
int      y, x1 = 60, x2 = 260, pixval = 1;

  if ((vidAdap == mda) || (vidAdap == other))         /* can't do it */
    return;                                            /* so go back */
  setmode (4);                           /* go to CGA graphics mode */
  gotoxy (8, 0, 0);
  puts ("320 x 200 color graphics");                    /* show mode */
  palette (0);
  for (y = 50; y < 151; y++ ) {                      /* draw figure */
    hdraw (x1, x2, y, pixval);
    x1 += 2; x2 -= 2;                                  /* change x's */
    if (y ==  84) pixval = 2;                      /* change colors */
    if (y == 117) pixval = 3;
  }
  wait ();
  setmode (vmode);
} /* ------------------------- */
```

End Listings

SAS Institute Inc. Announces

Lattice C Compilers for Your IBM Mainframe

Two years ago...

SAS Institute launched an effort to develop a subset of the SAS® Software System for the IBM Personal Computer. After careful study, we agreed that C was the programming language of choice. And that the Lattice® C compiler offered the quality, speed, and efficiency we needed.

One year ago...

Development had progressed so well that we expanded our efforts to include the entire SAS System on a PC, written in C. And to insure that the language, syntax, and commands would be identical across all operating systems, we decided that all future versions of the SAS System—regardless of hardware—would be derived from the same source code written in C. That meant that we needed a C compiler for IBM 370 mainframes. And it had to be good, since all our software products would depend on it.

So we approached Lattice, Inc. and asked if we could implement a version of the Lattice C compiler for IBM mainframes. With Lattice, Inc.'s agreement, development began and progressed rapidly.

Today...

Our efforts are complete—we have a first-rate IBM 370 C compiler. And we are pleased to offer this development tool to you. Now you can write in a single language that is source code compatible with your IBM mainframe and your IBM PC. We have faithfully implemented not only the language, but also the supporting library and environment.

Features of the Lattice C compiler for the 370 include:

- **Generation of reentrant object code.** Reentrancy allows many users to share the same code. Reentrancy is not an easy feature to achieve on the 370, especially if you use non-constant external variables, but we did it.
- **Optimization of the generated code.** We know the 370 instruction set and the various 370 operating environments. We have over 100 staff years of assembler language systems experience on our development team.
- **Generated code executable in both 24-bit and 31-bit addressing modes.** You can run compiled programs above the 16 megabyte line in MVS/XA.
- **Generated code identical for OS and CMS operating systems.** You can move modules between MVS and CMS without even recompiling.
- **Complete libraries.** We have implemented all the library routines described by Kernighan and Ritchie (the informal C standard), and all the library routines supported by Lattice (except operating system dependent routines), plus extensions for dealing with 370 operating environments directly. Especially significant is our byte-addressable Unix®-style I/O access method.
- **Built-in functions.** Many of the traditional string handling functions are available as built-in functions, generating in-line machine code rather than function calls. Your call to move a string can result in just one MVC instruction rather than a function call and a loop.

In addition to mainframe software development, you can also use our new cross-compiler to develop PC software on your IBM mainframe. With our cross-compiler, you can compile Lattice C programs on your mainframe and generate object code ready to download to your PC.

With the cross-compiler, we also offer PLINK86™ and PLIB86™ by Phoenix Software Associates Ltd. The Phoenix link-editor and library management facility can bind several compiled programs on the mainframe and download immediately executable modules to your PC.

Tomorrow...

We believe that the C language offers the SAS System the path to true portability and maintainability. And we believe that other companies will make similar strategic decisions about C. Already, C is taught in most college computer science curriculums, and is replacing older languages in many. And almost every computer introduced to the market now has a C compiler.

C, the language of choice...

C supports structured programming with superior control features for conditionals, iteration, and case selection. C is good for data structures, with its elegant implementation of structures and pointers. C is conducive to portable coding. It is simple to adjust for the size differences of data elements on different machines.

Continuous support...

At SAS Institute, we support all our products. You license them annually; we support them continuously. You get updates at no additional charge. We have a continuing commitment to make our compiler better and better. We have the ultimate incentive—all our software products depend on it.

For more information...

Complete and mail the coupon today. Because we've got the development tool for your tomorrow.

SAS®

SAS Institute Inc.
SAS Circle, Box 8000
Cary, NC 27511-8000
Telephone (919) 467-8000 x 7000

I want to learn more about:

☐ the C compiler for MVS software developers
☐ the C compiler for CMS software developers
☐ the cross-compiler with PLINK86 and PLIB86

today...so I'll be ready for tomorrow.

Please complete or attach your business card.

Name ______________________
Title ______________________
Company ______________________
Address ______________________
City ______________ State ________ ZIP ________
Telephone ______________________

Mail to: SAS Institute Inc., Attn: CC, SAS Circle, Box 8000, Cary, NC, USA. 27511-8000. Telephone (919) 467-8000, x 7000

DDJ 1187

Listing One *(Text begins on page 46.)*

```
Screen # 0

\                                                    jbb 15:47 08/04/87

                              GETIMAGE


                 Copyright 1987, All Rights Reserved

                                  by
                          J. Brooks Breeden
                           Columbus, Ohio
                           August 4, 1987

            Loads an EGAPAINT file into video memory
       from within UR/FORTH for subsequent animation, etc.



Screen # 1

\ Misc.                                              jbb 13:01 08/04/87
DOSINT                                   \ fetch level 2 DOS interface
HCB IMAGEHCB                             \ create the handle block
CREATE PALTAB  16 ALLOT                  \ to hold egapaint's palette

: ?MEMREQ ( x1 y1 x2 y2 - bytes)  \ calc. memory req to @BLOCK
   ROT - 1+  -ROT SWAP - 1+ 2/ 1+  *  4 + ;

HEX
: RESTORE-EGA ( - )            \ restore EGA's default mode:
   2 3C4 PC!  0F 3C5 PC!       \ default map mask
   3 3CE PC!   0 3CF PC!       \ default data rotate reg value
   5 3CE PC!   0 3CF PC!       \ default write mode 0
   8 3CE PC!  FF 3CF PC! ;     \ default bit mask
DECIMAL
-->


Screen # 2

\ EGA map mask control                               jbb 15:37 08/04/87

HEX

: READMODE0 ( - )   5 3CE PC!  8 3CF PC! 2 3CE PC! 0 3CF PC! ;

          \ set map-masks to select active EGA color bit-planes
: BLUEPLANE     ( - ) 2 3C4 PC!  1 3C5 PC! ; \ plane 0
: GREENPLANE    ( - ) 2 3C4 PC!  2 3C5 PC! ; \ plane 1
: REDPLANE      ( - ) 2 3C4 PC!  4 3C5 PC! ; \ plane 2
: INTENSEPLANE ( - ) 2 3C4 PC!  8 3C5 PC! ; \ plane 3

DECIMAL
-->



Screen # 3

\ File control                                       jbb 12:59 08/04/87

: OPENIMAGEFILE ( ^filename - )
   IMAGEHCB NAME>HCB                          \ force filename
   IMAGEHCB O_RD FOPEN                        \ open file for reading
   IF ABORT" Can't open file." THEN ;

: CLOSEIMAGEFILE ( - )  IMAGEHCB FCLOSE DROP ;

: MAKEIMAGEFILE ( ^filename - )
   IMAGEHCB NAME>HCB                          \ force filename
   IMAGEHCB 0 FMAKE                           \ make "normal" file
   IF ABORT" Can't make file." THEN ;
-->



Screen # 4

\ Read EGAPAINT file from disk to video      jbb 15:45 08/04/87

\ To load EGAPAINT disk image into EGA video segment...
\ set map mask, read bytes from disk, repeat for each plane

: READBYTES ( - )      \ read  28,000 bytes to video segment
   IMAGEHCB  ?VSEG 0  \ source-handle; dest.- videoseg:offset
   28000 FREADL        \ read 28,000 bytes
   DROP ;              \ drop flag.

: READPAINTFILE ( - ) \ read all four planes in sequence...
     BLUEPLANE  READBYTES  REDPLANE READBYTES
     GREENPLANE READBYTES  INTENSEPLANE READBYTES ;
-->



Screen # 5

\
                                                     jbb 15:44 08/04/87
: LOADPAINTFILE ( ^filename - )
   640X350 VMODE  CLS                 \ hi-res 16-color
   OPENIMAGEFILE                         \ Open the file, and
   IMAGEHCB PALTAB 16 FREAD              \ read palette data.
     IF PALTAB !PALETTE                  \ If read was successful,
      THEN                               \ set palette to new colors.
   READMODE0                             \ Set read mode 0 and
   READPAINTFILE                         \ read the bit-plane data.
   CLOSEIMAGEFILE                        \ Close the file, and
   RESTORE-EGA ;                         \ restore the EGA defaults.
-->



Screen # 6

\ Load the EGAPAINT full-screen image          jbb 15:17 08/04/87

CREATE BARON  ,C" THEFOKKR.IMG"  \ orig 112K EGAPAINT disk file
CREATE FOKKER ,C" DERFOKKR.IMG"  \ new smaller @BLOCK disk file
CREATE DR.1   3072 ALLOT         \ memory req as multiple of 8

: SAVEDR.1    1 1 106 53  DR.1 @BLOCK ;    \ get block to memory

: GETIMAGE BARON LOADPAINTFILE SAVEDR.1 ; \ get orig image>mem

: SAVEFOKKER   FOKKER MAKEIMAGEFILE  \ build .img file on disk
   IMAGEHCB PALTAB 16 FWRITE DROP    \ include pallete info
   IMAGEHCB DR.1 2904 FWRITE DROP    \ write DR.1 mem to disk
   CLOSEIMAGEFILE ;                  \ close it out...
-->


Screen # 7

\ CHECKOUT  JUNK                                jbb 16:07 08/04/87

: CK1  HR DR.1 267 148 !BLOCK
       277 156 361 192 FRAME           \ Fokker image limits
       LTBLUE FG
       267 148 372 200 FRAME           \ image area saved
       262 143 272 153 FRAME           \ hit zone around CORNER
       314 165 324 175 FRAME ;         \ hit zone on Fokker

: GUNSIGHT ( - )   ORANGE FG  269 170 369 170 LINE
   319 133 319 274 LINE  319 170 50 circle ;

: CK  CK1  GUNSIGHT ;
```

End Listing One

Listing Two

```
Listing 2.  Pseudocode to move a full 112,000 byte image
from the disk to the screen:  NOTE:  ALL NUMBERS HEXIDECIMAL

Set the video mode to graphics mode xx however your language
does it.

Open the file using a DOS handle however your language does
it.

If palette data is the first data in the file,
Then read the palette data and set the new palette.  (This
should be covered in your language's high level routines.)

Set read mode 0
Write  5 to port 3CE, then
write  8 to port 3CF, then
write  2 to port 3CE, then
write  0 to port 3CF.

Set map mask to plane 0 (blue plane)
Write  2 to port 3C4, then
write  1 to port 3C5.

Read 28,000 bytes from the file to A000:0000 however your
language does it.

Set map mask to plane 2 (red plane)
Write 2 to port 3C4, then
write 4 to port 3C5.

Read 28,000 bytes from the file to A000:0000 however your
language does it.

Set map mask to plane 1 (green plane)
Write 2 to port 3C4, then
write 4 to port 3C5.

Read 28,000 bytes from the file to A000:0000 however your
language does it.

Set map mask to plane 3 (intensity plane)
Write  2 to port 3C4, then
write  8 to port 3C5.

Read 28,000 bytes from the file to A000:0000 however your
language does it.

Close the file, however your language does it.

Restore EGA default values.
(set map mask default: all planes active)
Write  2 to port 3C4, then
write 0F to port 3C5.
```

```
(set data rotate register default)
Write  3 to port 3CE, then
write  0 to port 3CF.

(set default write mode 0)
Write  5 to port 3CE, then
write  0 to port 3CF.

(set default bit-mask)
Write  8 to port 3CE, then
write FF to port 3CF.
```

End Listing Two

Listing Three

```
Screen # 0

\                                                 jbb 17:28 07/04/87

                          FOKKER.SCR
                A Mindless Game of Motor Skill
                (somewhere over the trenches...)

NOTE: YOU ABSOLUTELY MUST HAVE CRUISE CONTROL, OR SOME OTHER
      KIND OF KEYBOARD SPEEDUP SOFTWARE FOR THIS TO WORK !!!

                       Standard Version
             Copyright (C) 1987, All Rights Reserved

                              by
                       J. Brooks Breeden

              EGAGRAPH.EXE must be loaded first


Screen # 1

\ File control...                                 jbb 13:08 08/04/87
\ this screen is required to load image initially.
\ For turnkey, the image is in memory & this scr isn't required

 DOSINT                              \ level 2 DOS interface

HCB IMAGEHCB                        \ create the handle block

: OPENIMAGEFILE ( ^filename - )
   IMAGEHCB NAME>HCB                \ force filename
   IMAGEHCB O_RD FOPEN              \ open file for reading
   IF ABORT" Can't open file." THEN ;

: CLOSEIMAGEFILE  ( -- )
   IMAGEHCB FCLOSE DROP ;           \ close, ignore status
-->


Screen # 2

\ File control...                                 jbb 09:49 08/01/87

CREATE DR.1   3000 ALLOT              \ to hold the bit map image
CREATE PALTAB   16 ALLOT              \ new palette of colors
CREATE FOKKERIMG ," DERFOKKR.IMG"     \ name of the image file

: GETFOKKER             \ load binary image from disk to memory
   DR.1 3000 ERASE                    \ clean out memory area
   FOKKERIMG OPENIMAGEFILE            \ open 'er up
   IMAGEHCB PALTAB 16 FREAD DROP      \ read in palette info
   IMAGEHCB DR.1 2866 FREAD  DROP     \ read in the image
   CLOSEIMAGEFILE ;                   \ close the file

GETFOKKER        \ load the Fokker image during the compile...
-->              \ Note: Obviously, you'd better have the image
                 \ file in your directory at this point!


Screen # 3

\ Pallette colors                                 jbb 17:10 08/02/87

: FG   FOREGROUND ;                  \  shorthand for LMI's words
: BG   BACKGROUND ;
: HR   640X350 VMODE ;               \ high res EGA 16-color mode

\ Named colors for the EGAPAINT default pallette
0 CONSTANT BLACK       8 CONSTANT GREEN
1 CONSTANT DKGRAY      9 CONSTANT LTGREEN
2 CONSTANT GRAY       10 CONSTANT CYAN
3 CONSTANT DKRED      11 CONSTANT LTBLUE
4 CONSTANT RED        12 CONSTANT BLUE
5 CONSTANT ORANGE     13 CONSTANT DKBLUE
6 CONSTANT YELLOW     14 CONSTANT PURPLE
7 CONSTANT DKGREEN    15 CONSTANT WHITE
-->


Screen # 4

\ Utilities                                       jbb 09:53 08/01/87

 : TAB ( row col - )  SWAP  GOTOXY ;            \ clearer for text
 : ?   @ . ;
                  \ returns unique IBM keycode for each keypress
 : MYKEY ( - n)   KEY DUP 0= IF DROP KEY 128 + THEN ;

 : WAIT ( - )   MYKEY  DROP ;                  \ wait for a keypress
 \ vector math...
 : V+ ( a b c d --- a+c b+d)   >R ROT + SWAP R> + ;
 : V* ( a b c d --- a*c b*d)   >R ROT * SWAP R> * ; \ not used;
 : V/ ( a b c d --- a/c b/d)   >R ROT / SWAP R> / ; \ for info..
: S-LINE ( row - )   0 TAB  80 0 DO 196 EMIT LOOP ; \ a line
: D-LINE ( row - )   0 TAB  80 0 DO 205 EMIT LOOP ; \ dbl. line
-->

Screen # 5

\ Random number generator                         jbb 09:56 08/01/87

VARIABLE SEED
@TIME COMBINE SEED !                    \ plant seed w/system clock

: random ( - n )                               \ 0 <= n <= 32767
   SEED @ 259 * 3 + 32767 AND  DUP SEED ! ;

: RANDOM ( n1 - n2 )                                \ 0 <= n2 < n1
   random M* 32768 UM/MOD NIP ;

: BETWEEN  ( lo# hi# - inbetween#)   OVER - RANDOM + ;

: WITHIN? ( n hi# lo# - flag)   >R 1- OVER < SWAP R> < AND ;
-->


Screen # 6

\ Fokker movement                                 jbb 14:44 08/06/87

2VARIABLE CORNER    \ xy-coords of up-left corner of fokker box

-1 1 2EQU RANGE                 \ range of random fokker movement

          \ add to Baron's excitement by bumping allowed RANGE
: EXCITEMENT ( - )   RANGE -1 1 V+ 2EQU RANGE ;

   \ leaves random x and y within RANGE by which to move Fokker
: MOVEFOKKER ( - n n)   RANGE BETWEEN RANGE BETWEEN ;

: GUNSIGHT ( - )   ORANGE FG  269 170 369 170 LINE
   319 133 319 274 LINE  319 170 50 circle ;
-->


Screen # 7

\ Stick control                                   jbb 10:05 08/01/87

                  \ set gunsight movement based on #key pressed
: ?stick ( adr - adr n n)  MYKEY
      CASE 199 OF  2 -2 ENDOF    200 OF  0 -2 ENDOF
           201 OF -2 -2 ENDOF    205 OF -2  0 ENDOF
           209 OF -2  2 ENDOF    208 OF  0  2 ENDOF
           207 OF  2  2 ENDOF    203 OF  2  0 ENDOF
     ( otherwise)  0  0              \ leave 0's for V+ to add...
           ROT                       \ move index to top of stack
     ENDCASE  ;                      \ for ENDCASE to drop...

: ?STICK ( adr - adr n n) ?TERMINAL   \ key pressed?
   IF  ?stick                         \ check stick control
   ELSE  0 0  THEN ;                  \ leave zeros for "V+"
-->

Screen # 8

\ Exclamations!                                   jbb 10:12 08/01/87

\ These aren't real serious curses; you make up your own!
CREATE CURSES ," Dankeschoen!        Welkommen!          Wie Geh
ts?          Was ist das?        Weiner Schnitzel!   Bratwurst!
       Sauerbrauten!        Knockwurst!          Sauerkraut!
    Weisswurst           "

: EXCLAMATION      \ shout a curse at the user, for distraction
   3 30 TAB CLREOL  3 33 TAB ORANGE FG
   10 RANDOM 20 * CURSES + 20  -TRAILING TYPE ;
-->


Screen # 9

\  Damage report                                  jbb 10:14 08/01/87

VARIABLE #HITS                           \ # of hits in cockpit zone
VARIABLE AMMO                            \ rounds of ammunition left

: .HITS   GRAY FG 22 44 TAB #HITS ? ;     \ print out number of
: .AMMO   GRAY FG 23 44 TAB AMMO  ? ;      \ hits & ammo left.

: HIT?   CORNER 2@ 143 153 WITHIN? \ y in hit zone?
   SWAP 262 272 WITHIN?  AND       \ x in hit zone?  Both?

    IF 2 #HITS +! .HITS               \ Increment hits & show #.
      #HITS @ 4 MOD 0=                \ For every 4 hits, increase
      IF EXCITEMENT  EXCLAMATION      \ movement & display curse!
      THEN
    THEN ;
 -->


 Screen # 10

 \  Shooting...                                   jbb 10:19 08/01/87

 : 2SHOTS   219 349 319 170 LINE 419 349 320 170 LINE ;

 : FIREGUNS     \ fire two shots, decrement ammo, show ammo left
    WHITE FG  2SHOTS
    5000 0 DO  LOOP   \ kill some time
    DKBLUE FG  2SHOTS  -2 AMMO +! .AMMO ;
```

(continued on next page)

Listing Three *(Listing continued, text begins on page 46.)*

```
: SHOOTING?   ?TERMINAL           \ If there is keyboard input...
   IF MYKEY 32 =                  \ was it the spacebar?
     IF AMMO @                    \ Yes, if we have any ammo,
       IF FIREGUNS HIT? THEN      \ shoot & check if we hit the
     THEN                         \ Fokker.
   THEN ;
-->


Screen # 11

\ Explosion                                     jbb 15:01 08/04/87

: BURST   320 0   \ random dots expanding in all four quadrants
   DO 319 I RANDOM + 174 I RANDOM - !PEL
      319 I RANDOM + 174 I RANDOM + !PEL
      319 I RANDOM - 174 I RANDOM + !PEL
      319 I RANDOM - 174 I RANDOM - !PEL LOOP  ;

: EXPLODE   3 0 DO  BURST  LOOP ;   \ it pulses the bursts...

HEX                     \ clear bios buffer of waiting keypresses
: CLR-KEY-BUF    0C01 regAX ! 21 INT86 ;
DECIMAL

: WAIT-FOR-ESC   CLR-KEY-BUF  BEGIN  MYKEY 27 =  UNTIL ;
-->


Screen # 12

\ Win message                                   jbb 17:35 07/04/87

: WIN                      \ what happens if you shoot him down!
   WHITE FG EXPLODE  DKBLUE FG REVERSE  ORANGE FG 6 14 TAB
   ."  What a pilot!  You downed the baron with "
   100 AMMO @ - .  ." rounds. "
   7 14 TAB
   ."  Now it's time to head for home and celebrate.       "
   20 25 TAB
   ."  Press ESC to go home in GLORY. " GRAY FG REVERSE
   WAIT-FOR-ESC ;
-->


Screen # 13

\ Lose message                                  jbb 10:20 08/01/87

: LOSE             \ what happens if you DON'T shoot him down...
   DKBLUE FG REVERSE  ORANGE FG 6 14 TAB
   ."  You are a rotten pilot!  You waste ammunition, and "
   7 14 TAB
   ."  let yourself get outflown by the bloody Red Baron. "
   8 14 TAB
   ."  Tuck your tail beween your legs and head for home. "
   20 23 TAB
   ."  Now press ESC to go home in SHAME! "
   DKBLUE FG REVERSE  GRAY FG  WAIT-FOR-ESC ;
-->


Screen # 14

\ Setup                                         jbb 10:22 08/01/87

F: FOKKER                                 \ forward reference

: SETUP   HR PALTAB !PALETTE CLS          \ use EGAPaint palette
    DKBLUE BG  GUNSIGHT                   \ paint sky and gunsight
   -1 1 2EQU RANGE                        \ fokker motion range
   175 375 BETWEEN  104 254 BETWEEN       \ random starting points
   CORNER 2!  0 #HITS !  100 AMMO !       \ setup the variables
   GRAY FG
   22 34 TAB ." # OF HITS: " .HITS         \ show hits
   23 34 TAB ." AMMO LEFT: " .AMMO ;       \ show ammo
-->

Screen # 15

\ Major loop                                    jbb 10:23 08/01/87

: DOGFIGHT   SETUP                      \ get ready...
   BEGIN  DR.1                          \ address of fokker bitmap
     CORNER 2@ MOVEFOKKER V+            \ +random movement
     ?STICK V+       ( --- adr x y)     \ +stick control
     2DUP CORNER 2! !BLOCK              \ update corner; show plane
     SHOOTING?  #HITS @ 19 >            \ check hits in pilot zone
       IF WIN  EXIT THEN                \ blow up plane; you win.
     AMMO @ 0=                          \ out of ammo?
       IF LOSE EXIT THEN                \ you lose...
     GUNSIGHT                           \ redisplay gunsight
   AGAIN  ;
-->

Screen # 16

\ Options
                                                jbb 10:23 08/01/87
: D-LINES
   B/W CLS PURPLE FG 7 D-LINE 19 D-LINE RED FG ; \ double lines

: .OPTIONS    D-LINES
    6  0 TAB ." Fokker" 6 36 TAB ." OPTIONS"    LTBLUE FG
   11 31 TAB ." F1.  " GRAY FG ." Instructions" LTBLUE FG
   13 31 TAB ." F2.  " GRAY FG ." Play"         LTBLUE FG
   15 31 TAB ." F10. " GRAY FG ." Exit"
   20 0 TAB ;
-->



Screen # 17

\ Credits                                       jbb 13:25 07/31/87

: .CREDITS    D-LINES  6 36 TAB ." FOKKER"  LTBLUE FG 8 23 TAB
   ." (A Mindless Game of Motor Skill)"  11 20 TAB  RED FG
   ." Somewhere over the trenches in France..."
   GRAY FG 14 22 TAB
   ." Copyright 1987, All Rights Reserved" 15 38 TAB ." by "
   16 32 TAB ." J. Brooks Breeden" 17 32 TAB ." Columbus, Ohio"
    ORANGE FG  20 26 TAB ." Press Any Key to Continue..."
   23 11 TAB LTBLUE FG
   ." Written in UR/FORTH from Laboratory Microsystems, Inc."
   24  9 TAB
 ." Fokker Dr.1 created with EGAPAINT from RIX Softworks, Inc."
  ;
-->


Screen # 18

\ Help...                                       jbb 13:24 07/31/87

: HELP1   B/W CLS RED FG  1 20 TAB ." Your Mission:"
   PURPLE FG 2 S-LINE  LTBLUE FG  3 20 TAB
   ." Your mission is to down the Red Baron with 20 hits"
   4 20 TAB
   ." in the cockpit area of his Fokker Dr.1 triplane."
   5 20 TAB
   ." When you shoot, the Baron gets excited, and takes"
   6 20 TAB
   ." evasive action.  You have 100 rounds of ammo left,"
   7 20 TAB
   ." and your guns have a tendency to jam..." ;
-->


Screen # 19

\ Help...                                       jbb 13:21 07/31/87

: HELP2   RED FG  9 20 TAB
   ." To play..."  PURPLE FG 10 S-LINE LTBLUE FG  11 20 TAB
   ." The cursor keys control the gunsight like a joystick."
   12 20 TAB
   ." The up arrow key pushes the stick forward.  The down"
   13 20 TAB
   ." arrow key pulls back on the stick.  Left, right, and"
   14 20 TAB
   ." diagonal keys move the stick as you would expect."
   16 20 TAB
   ." Fire by holding down the space bar."
   21 20 TAB
   ." Press Ctrl-Break to quit at any time." ;
-->

Screen # 20

\ Warning...                                    jbb 10:24 08/01/87

: WARNING    D-LINES  6 34 TAB ." WARNING!!!" LTBLUE FG
    9 20 TAB ." YOU ABSOLUTELY MUST HAVE CRUISE CONTROL,"
   10 20 TAB ." QUICKEYS, OR OTHER TSR KEYBOARD SPEED-UP"
   11 20 TAB ." RESIDENT IN ORDER TO RUN THIS PROGRAM !!!"
   13 20 TAB ." If you do not have a keyboard speed-up"
   14 20 TAB ." program resident, press Ctrl-Break to exit."
   16 20 TAB ." If a keyboard speed-up program is resident..."
   ORANGE FG  20 26 TAB ." Press Any Key to Continue..."
   WAIT ;

: .HELP   HELP1 HELP2  ORANGE FG  23 26 TAB
   ." Press Any Key to Continue..."  WAIT FOKKER ;
-->


Screen # 21

\ Main resolved                                 jbb 10:24 08/01/87

: INTRO   CLS B/W  .CREDITS WAIT  20 0 TAB CLREOL ;

R: FOKKER .OPTIONS MYKEY       \ resolve FOKKER forward reference
   CASE  187 OF .HELP FOKKER ENDOF          \ show instructions
         188 OF  DOGFIGHT  FOKKER ENDOF     \ do the dogfight
         196 OF  BYE  ENDOF                 \ exit to DOS
         DROP FOKKER                        \ recurse to options
   ENDCASE ;


: MAIN   HR PALTAB !PALETTE WARNING INTRO FOKKER ;
```

End Listings

DIAGNOSTICS

for Personal Computers from *Windsor Technologies*

Windsor Technologies offers a family of diagnostic products for IBM PC, PC XT, PC AT and compatible computers. Originally designed for and used by service industry professionals, ***Windsor's*** products are well suited for anyone involved in the management and maintenance of personal computers.

PC-Technician™

Professional Level Personal Computer Diagnostic System

PC-Technician is an 'advanced' type diskette-based diagnostic system that provides comprehensive testing for each individual system component. Information about the testing and the results are presented via text and graphics on the system's display. ***PC-Technician*** provides on-line help displays and information about the system's configuration.

PC-Technician is ideal for service personnel, manufacturers, OEM's, and users who need detailed information about the status of their computer's hardware.

PC-Technician is available in the following configurations:

- **Product Code T01 (for PC, PC XT)** **$195.00**
- **Product Code T51 (for PC AT)** **$245.00**
- **Product Code T90 (for PC, PC XT, PC AT—** T01 and T51 in the same carrying case**)** **$395.00**

A FREE Evaluation Diskette is available—see the coupon below.

T90—PC, PC XT, PC AT

WindsorPOST™

Personal Computer Diagnostic ROM Module

WindsorPOST is designed to be used when the system is malfunctioning such that its 'power-on-self-test' (commonly called POST) does not complete satisfactorily and a diskette-based diagnostic product, such as PC-Technician, cannot be loaded and used for troubleshooting.

WindsorPOST temporarily replaces the system's BIOS ROM module. When installed, its LED indicates if the proper amount of power is at the BIOS ROM socket. It then performs 36 tests on the circuits and electrical components on the system board, the system memory up to 640K, and the color or monochrome video display memory. Errors are identified down to the module (chip) level in three full displays of information on the system's video display unit.

WindsorPOST is intended for those service operations where specific identification of failing components is required and where soldered components are removed and replaced. ***WindsorPOST*** has been in use in the field for over one year and performance data indicates that it properly identifies the failing component at least 70% of the time.

WindsorPOST is currently available for use on IBM PC's and PC XT's (a PC AT type product is under development) and those compatible systems where the system board circuitry and components are extremely similar to the IBM standard. ***WindsorPOST*** comes with a 30-day money back guarantee.

- **Product Code P01 (PC, PC XT)** **$295.00**

P01—PC, PC XT

For immediate ordering, contact ***Windsor Technologies*** at **(415) 456-2209,** 8:30 AM-5:00 PM Pacific Coast time. VISA, Mastercard, American Express and COD orders accepted.

Volume discounts and reseller discounts are available.

Windsor Technologies, Inc.
130 Alto Street
San Rafael, CA 94901
(415) 456-2200

PC-Technician and WindsorPOST are trademarks of ***Windsor Technologies, Inc.***, and IBM PC, PC XT, PC AT are trademarks of International Business Machines Corporation.

Windsor Technologies, Inc.
130 Alto Street, San Rafael, CA 94901 · (415) 456-2200

Please Send A Free ***PC-Technician*** Evaluation Diskette ☐

Please Send Info About ***PC-Technician*** ☐ ***WindsorPOST*** ☐

NAME__________ TITLE__________

COMPANY__________ (TELEPHONE)__________

ADDRESS__________

CITY__________ STATE______ ZIP______

☐ SERVICE CO. ☐ OEM ☐ MANUFACTURER ☐ DEALER ☐ VAR/VAD

☐ IN-HOUSE SERVICE ☐ USER OTHER__________

DDJ 1187

BASIC

↓

C or Pascal

translator

Ready to run, Syntax error free code
Structured, Indented, Scoped
Random file, Grapics are supported
Free Basic Runtime Library(Source code)
200 Page manual,Demo disk
BAS_C for Microsoft,Turbo,Lattice,Instant,etc
BAS_PAS for Turbo Pascal

Gotoless Conversion
Box 835910,Richardson,TX75083
Phone(214)404-1404

CIRCLE 116 ON READER SERVICE CARD

CANADA'S SOURCE FOR C

- Canadian Sales
  - Canadian Service
    - Canadian Technical Support
      - Canadian Product Knowledge

We specialize in programming & development software

LIFEBOAT • LATTICE • GREENLEAF • PHOENIX
SOFTCRAFT • MICROSOFT • BLAISE • ESSENTIAL
AGE OF REASON • DESMET • AZTEC
MARK WILLIAMS • GIMPEL • ROUNDHILL • GSS
HALO • FAIRCOM • RAIMA • INTEL • etc. • etc. •

VISA Call for full price list—Dealer enquiries welcome MasterCard

We know our products—we use them!

SCANTEL SYSTEMS LTD.
801 York Mills Rd., Don Mills, Ont., M3B 1X7
(416) 449-9252

CIRCLE 391 ON READER SERVICE CARD

Listing One

(Text begins on page 38.)

```
    @Listing 1@.  The @SaveRegion@ Procedure copies any
    upright rectangular graphics screen region into
    a buffer, @buff@, where its upper-left (x1,y1)
    and lower-right (x2,y2) screen coordinates are
    specified.  GETMEM is used to allocate memory,
    rather than the standard NEW, because the buffer
    size needs to be computed at run time.


{The following code is the property of H.D.
Callihan, University of Pittsburgh at Johnstown,
Johnstown, Pa.  Personal use is encouraged.  Feel
free to make copies for distribution to other
personal users.  Commercial use is prohibited
without written permission and an
appropriate license.

Version:  4      (requires the CRTmode function in
                 file:  CRTMODE.INC) Purpose:
                 save and restore a current screen
                 region in low- or hi-res mode.
   Date:  6/22/87
 Author:  H.D. Callihan, Ph.D.  (C) Copyright 1987
 Applic:  Turbo V3.0 for IBM PC and true compatibles.
   File:  GREGION4.INC
}


TYPE
    buffermemory = ARRAY[1..3] OF INTEGER;
    bufferaddress = ^buffermemory;

PROCEDURE SaveRegion(VAR buff : bufferaddress;
                         x1,y1,  {upper left}
                         x2,y2   {lower right}
                               : INTEGER);

{---------- Local functions to SaveRegion-----------}

   FUNCTION max(a,b: INTEGER) : INTEGER;
      BEGIN
         IF a<b THEN max := b  ELSE max := a
      END;

   FUNCTION min(a,b : INTEGER) : INTEGER;
      BEGIN
         IF a<b THEN min := a  ELSE min := b
      END;

{---------- End local functions --------------------}

VAR   width, height, size : INTEGER;
          dummy1, dummy2 : BYTE;

BEGIN  {SaveRegion}

  {correct for negative x and y}
  x1 := max(x1,0); x2 := max(x2,0);
  y1 := max(y1,0); y2 := max(y2,0);

  {correct for large y}
  y1 := min(y1,199);  y2 := min(y2,199);

  {compute height of image in pixels}
  height := ABS(y1-y2) +1;

  CASE CRTmode(dummy1, dummy2) OF
           { dummy1 and dummy2 not used }
   4,5: {one of the low resolutions}
       BEGIN
          x1 := min(x1,319);  x2 := min(x2,319);

          {compute width of image in pixels}
          width := ABS(X1-X2) +1;

          {compute size of buffer need to store image}
          size := ((width+3) DIV 4) * height * 2  + 6;

          GETMEM(buff, size);
          GETPIC(buff^, x1,y1,x2,y2)
       END;

    6: {high resolution}
      BEGIN
          x1 := min(x1,639);  x2 := min(x2,639);
          width := ABS(x1-x2) +1;
          size := ((width+7) DIV 8) * height  + 6;
          GETMEM(buff, size);
          GETPIC(buff^, x1,y1,x2,y2)
      END;
    ELSE WRITE(^G); {unacceptable mode}
  END  {CASE}
END;  {SaveRegion}
```

End Listing One

Listing Two

```
    @Listing 2@.  The @CRTmode@ function determines which
    display mode is currently active.  It returns an integer
    code as well as two arguments which determine text row
    and column information if a text mode is active.
```

```
FUNCTION CRTmode(VAR  char_columns,
                      display_page : BYTE) : BYTE;

 {Returns CRT mode of operation.  It uses registers and
 software interrupt 10h to BIOS video services.  Also
 returns the number of character columns in the current
 video mode (80 or 40) and the video display page as
 VARiable parameters.

Version:  2
 Author:  H. D. Callihan, PhD,  UPJ
   Date:  2/29/87
   File:  CRTMODE.INC

VIDEO MODES:
     0 = 40x25 monochrome
     1 = 40x25 color
     2 = 80x25 monochrome
     3 = 80x25 color
     4 = 320x200 color graphics (40x25 text)
     5 = 320x200 mono graphics  (40x25 text)
     6 = 640x200 b/w high res graphics (80x25 text)

 The following values are returned on the AT&T 6300 for
 superres mode (640x400).

  $40 = 640x400 mono superres graphics
        (80x25 high quality text)

  $48 = 640x400 mono superres graphics
        (80x50 tiny text)

 Register input :   (AH) <-- 0Fh
 Register output:   (AL) --> current mode
                    (AH) --> number of character
                             columns on screen
                    (BH) --> current active display page

 Uses software interrupt 10h for BIOS video service.
 }

  TYPE regpack = RECORD
          ax,bx,cx,dx,bp,si,di,ds,es,flags: INTEGER
       END {regpack};

  VAR  dosreg : regpack;

  BEGIN       {CRTmode}
     WITH dosreg DO
        BEGIN
          {set high byte $0F for register input}
          ax := $0F00;
               { $0F = 00001111 binary}

          INTR($10, dosreg);   {software interrupt 10h}
          CRTmode := LO(ax);   {mask low byte}
          char_columns := HI(ax); {mask high byte}
          display_page := HI(bx)  {mask high byte}
        END
  END;       {CRTmode}
```

End Listing Two

Listing Three

```
 @Listing 3@.  The @RestoreRegion@ procedure copies any
 upright rectangular graphics screen region from a
 buffer, @buff@, previously saved by @SaveRegion@ where
 its lower-left screen coordinates (x,y) are specified.
 FREEMEM is used, rather than the standard DISPOSE,
 because the buffer size needs to be computed at run
 time.


PROCEDURE RestoreRegion
 (VAR buff : bufferaddress; {pointer}
          x,y : INTEGER;    {lower left corner}
       freeup : BOOLEAN);   {freemem after restore}

VAR width, height, resolution, size : INTEGER;
                          x_ok, y_ok : BOOLEAN;
BEGIN
   resolution := buff^[1];
   width := buff^[2];
   height := buff^[3];

   {check screen boundary y limits}
   y_ok := (y-height+1 >= 0) AND (y < 200);

   CASE resolution OF
     2: {low}
         BEGIN
          {check x limits}
          x_ok := (x >= 0) AND (x+width-1 < 320);

          IF x_ok AND y_ok THEN
          BEGIN
             PUTPIC(buff^,x,y);
             IF freeup THEN
             BEGIN
                size :=
                   ((width+3) DIV 4) * height * 2 + 6;
                FREEMEM(buff, size)
```

(continued on next page)

Hard Locks for Soft Parts.

At Rainbow Technologies, we think protecting software developers' investments is very serious business. That's why we designed the first fully effective security solution for software running on PCs and other computers.

Our family of virtually impenetrable Software Sentinel hardware keys provides the highest level of software protection the developer can get. While remaining invisible to the end user.

Take a look.

Key Sentinel Family Features.

Prohibits unauthorized use of software ▫ No need for copy protection ▫ Unlimited backup copies ▫ Virtually unbreakable ▫ Pocketsize key ▫ Transparent operation ▫ Transportable

Software Sentinel.

- Runs under DOS and Xenix, on IBM PC/XT/AT and compatibles
- Algorithm technique (Never a fixed response)
- Serial or parallel port version
- Minimal implementation effort
- Higher level language interfaces included
- 100 times faster than fixed-response devices (1ms)

Software Sentinel-C.

- For developers who want to customize or protect multiple packages with one device
- 126 bytes of non-volatile memory that is programmed before shipment of software
- We supply a unique programming adapter for programming the unit
- Higher level language interfaces included
- Runs under DOS on PC/XT/AT and compatibles
- Parallel port version only

Software Sentinel-W.

- Designed for workstations, supermicros and minicomputers
- Serial port only (modem-type)
- Algorithm technique
- We provide detailed interface specifications: Developer creates a port driver
- Interface requirements: 25 pin DB25P or DB25S; RS232/RS422/RS423
- Only signals used: DTR & RTS from computer; signal ground; DSR or optional DCD from Software Sentinel-W or external device. TXD, RXD, CTS, RI passed through.

Call For Software Sentinel Evaluation Kit Pricing.
International Distributors Wanted

RAINBOW TECHNOLOGIES
18011-A MITCHELL SOUTH IRVINE, CA 92714 USA
(714) 261-0228 TELEX: 386078 FAX: (714) 261-0260

CIRCLE 255 ON READER SERVICE CARD

Listing Three

(Listing continued, text begins on page 38.)

```
               END
            END
         END;
     1: {high}
         BEGIN
            x_ok := (x >= 0) AND (x+width-1 < 640);
            IF x_ok AND y_ok THEN
            BEGIN
               PUTPIC(buff^,x,y);
               IF freeup THEN {free the buffer}
               BEGIN
                  size :=
                     ((width+7) DIV 8) * height  + 6;
                  FREEMEM(buff, size)
               END
            END
         END;
     ELSE WRITE(^G^G)
   END  {CASE}
END;   {RestoreRegion}
```

End Listing Three

Listing Four

```
  @Listing 4@.  The @FreeBuffer@ procedure permits memory to
  be freed dynamically without displaying the image on the
  screen.  Like @RestoreRegion@, it also requires that the
  size of the memory block be computed based upon the
  block previously saved by @SaveRegion@.

PROCEDURE FreeBuffer ( VAR buff : bufferaddress );

VAR resolution, width, height, size : INTEGER;

BEGIN
   resolution := buff^[1];
   width := buff^[2];
   height := buff^[3];

   CASE resolution OF
      2 : {LOW}
          size := ((width+3) DIV 4) * height * 2 + 6;
      1 : {HIGH}
          size := ((width+7) DIV 8) * height  + 6;
      ELSE WRITE(^G^G^G)
   END; {CASE}
   IF resolution IN [1,2] THEN FREEMEM(buff, size)
END;  {FreeBuffer}
```

End Listing Four

Listing Five

```
  @Listing 5@.  The procedure, @SaveBlockToDisk@, facilitates
  saving a screen region to disk that was previously
  stored in a memory buffer using @SaveRegion@.  The file
  name is passed into the procedure as a string argument.
  @GetBlockFromDisk@ is a procedure which does the opposite
  by retrieving a block from disk and placing it into
  contiguous memory located at @buffer@.  @Resolution@ and
  @size@ are also computed and returned as a matter of
  convenience.  This recovered region may now be placed
  onto the screen using @RestoreRegion@.

{H D CALLIHAN, UNIVERSITY OF PGH AT JOHNSTOWN, 1987}

TYPE FileString80 = STRING[80];
PROCEDURE SaveBlockToDisk
   ( FileName : FileString80;
       buffer : bufferaddress  {mem address of block}
   );

VAR
     size, resolution, width, height : INTEGER;
     FileVariable : FILE;               {untyped file}

BEGIN
   resolution := buffer^[1];
   width := buffer^[2];
   height := buffer^[3];

   CASE resolution OF
      1 :  {HIGH}
           size := ((width + 7) DIV 8) * height + 6;
      2 :  {LOW}
           size := ((width + 3) DIV 4) * height *2 + 6;
      ELSE WRITE (^G^G^G^G)
   END:  {CASE}

   IF resolution IN [1,2] THEN BEGIN
```

M Street Software

80386 Support!

SCRUTINY

Advanced symbolic debugger.

- Multi-language: compatible with Turbo Pascal, Microsoft Assembler, others.
- Multi-DOS: works with all MS-DOS/PC-DOS computers.
- Multi-level: debug at source level and machine level, separately or together.
- Multi-display: debug character-mode and graphics-mode programs, with movable debug windows.
- Multi-chip: support for 8086, 80186, 80286, 80386.
- Fast 80386 "memory breakpoints" (stop program when specified variable is accessed or modified).

Scrutiny/Master $99.95
for debugging Turbo Pascal, Microsoft Assembler, and other languages.

Scrutiny/Turbo Special price! $49.95
for debugging Turbo Pascal only.

VISA/MC AMEX accepted. In Texas please add sales tax. Outside of North America add $10 per item shipping.

M Street Software
5400 E. Mockingbird Lane Suite 114
Dallas, Texas 75206
214-827-4908
Information also available via our 24 hour 300/1200 modem: 214-669-1882.

CIRCLE 275 ON READER SERVICE CARD

SAVE TIME and ADD PRODUCTIVITY with

XO-SHELL™

While you are in the MIDDLE of PROGRAMMING, with XO-SHELL you can

- VIEW ANY FILE, and transfer ANY SECTION to the PRINTER or into your EDITOR
- Do SOURCE-LISTING while you are in your application
- Do the MOST EFFICIENT CROSS-REFERENCING without leaving your editor
- OBTAIN KEY CODES without a reference or going through difficult interpretation
- INSERT GRAPHICS CHARACTERS in your source code
- View, copy and erase DIRECTLY from a SCROLLABLE DIRECTORY DISPLAY, plus
- DOS COMMAND EDITOR lets you retrieve, edit and reexecute previous DOS commands. It also PREVENTS ACCIDENTAL HARD DISK FORMATTING.

To order XO-SHELL for PC/MS-DOS, send $69 plus $5 shipping and handling, or call

1-800-635-5011 (outside MA)

WYTE™

WYTE CORPORATION
701 Concord Avenue
Cambridge, MA 02138
Tel. (617) 868-7704

CIRCLE 202 ON READER SERVICE CARD

```
      ASSIGN ( FileVariable, FileName );
      REWRITE ( FileVariable );
      BLOCKWRITE ( FileVariable, buffer^, 1+(size-1) DIV 128 );
      CLOSE ( FileVariable )
    END {IF}
  END;  { BlockSave }


PROCEDURE GetBlockFromDisk
    ( FileName    : FileString80;
  VAR buffer      : bufferaddress;
  VAR resolution,  {1=hi, 2=lo}
      size         : INTEGER );

VAR
   FileVariable : FILE;
   width, height,
   SizeOfFile    : INTEGER;
      {number of 128-byte records in block file}

BEGIN
   ASSIGN ( FileVariable, FileName );
   RESET ( FileVariable );
   SizeOfFile := FILESIZE ( FileVariable );

   IF SizeOfFile <> 0 THEN BEGIN
      GETMEM( buffer, SizeOfFile * 128 );
      BLOCKREAD ( FileVariable, buffer^, SizeOfFile );
      resolution := buffer^[1];
      width := buffer^[2];
      height := buffer^[3];
      CASE resolution OF
        1 : {HIGH}
            size := ((width+7) DIV 8)*height   +6;
        2 : {LOW}
            size := ((width+3) DIV 4)*height*2 +6;
        ELSE WRITE(^G^G^G^G^G)
      END {CASE}
  END;  {IF}
  CLOSE ( FileVariable );
END;    {GetBlock}
```

End Listing Five

Listing Six

```
    @Listing 6@.  The following program demonstrates the use
    of @SaveRegion@ and @RestoreRegion@.  Figure 2 contains
    images saved in @buffer@ and @buffer2@.


PROGRAM Test_SaveRegion_and_RestoreRegion;

{$I graph.p}       {-->    extended Turbo Graphics}

{$I crtmode.inc}   {-->    Callihan function to check
                           current crt mode}
{$I gregion4.inc}  {-->    Callihan save and restore
                           region Turbo Code
                           and FreeBuffer code}

VAR         i  : INTEGER;
    buffer,
    buffer2    : bufferaddress;
      {TYPE declared in the above Callihan file.}

      {The TYPE must be used here for SaveRegion,
      RestoreRegion, and FreeBuffer to work.    }

BEGIN
   GRAPHCOLORMODE;              {Pick your resolution}
{   HIRES; }   { use only colors 0 and 1 if this is used}

   GRAPHWINDOW(50,100,200,180);
        {pick window: (50,100) to (200,180)}

   FILLSCREEN(2);
           {clear current window to green=1, black=0,etc.}
                                       {red=2,  yellow=3 }

{Now, draw lines in black=0, green=1, etc.}
{draw a nice border around the window}
   DRAW(148,78,148, 2, 3);
   DRAW(148, 2,  2, 2, 3);
   DRAW(  2, 2,  2,78, 3);
   DRAW(  2,78,148,78, 3);

{Now, let's draw a couple circles in the window
 with center and radius determined by a loop index.
 Use color as last parameter.}

   FOR i := 10 TO 25 DO CIRCLE(3*i, 2*i, i,  1);

   READLN;      {wait for user to press return}

   SaveRegion(buffer, 0,0,150,80);
        {saves in current window coords}
        {151 pixels wide by 81 high    }

{    GRAPHCOLORMODE;}   {this call clears and resets
                         window to full screen}

   FILLSCREEN(0);       {clear window to black,
                         don't reset to full screen}
```

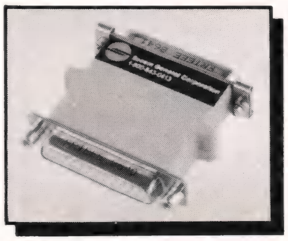
(continued on next page)

Would you like copy protection and customer satisfaction?

There's a better way to protect your software. It's called the Secom Key, and it works.

- ❑ *The Key is completely transparent to the end user.*
- ❑ *Won't interfere with peripheral operations.*
- ❑ *Doesn't occupy the disk drive.*
- ❑ *The Key allows unlimited backup copies.*
- ❑ *Makes site licensing easy and auditable.*
- ❑ *Easily installed. Uses only 1000 bytes.*
- ❑ *Over 60,000 have been sold worldwide.*
- ❑ *Same size as RS-232 plug.*
- ❑ *Available in quantities for as low as $19.95.*

For more information, contact Secom Information Products Co.



500 Franklin Square
1829 East Franklin Street
Chapel Hill, NC 27707

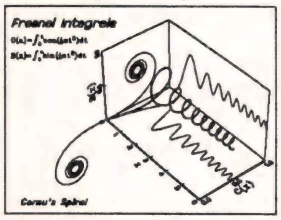
The Secom Key... for real software protection.

Secom Information Products Company
A Subsidiary of Secom General Corporation
Call Toll-free 1-800-843-0413

CIRCLE 394 ON READER SERVICE CARD

GraphiC™

can plot your data in publication quality



on your IBM PC

- linear, log, polar plots
- bar charts, Smith charts
- 3D curves, 3D surfaces
- 6 curve types, 8 markers
- 14 fonts, font editor
- 4096 x 3120 resolution
- zoom, pan, window plots
- high resolution printer and plotter dumps *in color*

Over 150 C and Assembler routines for full control

$395 with source code

For *personal* use only

VTEK™

DEC™ VT100/VT52 and Tektronix™ 4010/4014/4105 Terminal Emulator

- 20 user-defined keys
- scroll back buffer
- hardware 132 columns
- Kermit and XMODEM
- up to 800x600 screen resolution on EGAs
- zoom, pan, window plots
- "*hot key*" to DOS
- enhanced keyboard support
- ANSII extensions to VT100 for multi-color text
- scrolling VT100 window on graphics screen

$150. Site and source code licenses available

B-EDIT™

Our new binary editor for programmers - $29

MOST HARDWARE IS SUPPORTED

Scientific Endeavors Corporation

Route 4, Box 79 Kingston, TN 37763 (615) 376-4146

CIRCLE 210 ON READER SERVICE CARD

TURBO PASCAL GRAPHICS

Listing Six

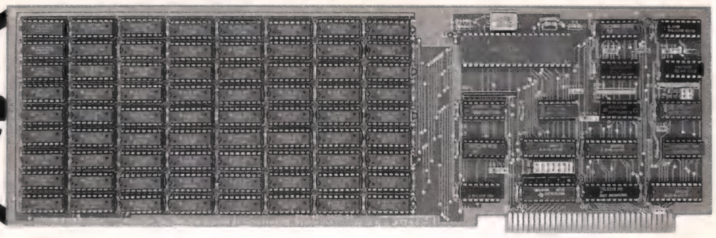
(Listing continued, text begins on page 38.)

```
READLN;

RestoreRegion(buffer,0,80,false);
     {Restores in current window coords }
     {at the lower left point 0,80      }
     {but does not free the buffer.     }

READLN;

GRAPHCOLORMODE;      {Pick resolution again}

{   HIRES;}

   RestoreRegion(buffer,0,80,false);

   READLN;

   FOR i := 1 TO 10 DO
        {let's get fancy}
        RestoreRegion(buffer,
                   150 - 8*i, 200 - 10*i, false);
   READLN;

   SaveRegion(buffer2,0,0,319,199);
           {Use buffer2 for background}

   FILLSCREEN(0);

   FOR i := 1 TO 10 DO BEGIN
      RestoreRegion(buffer, 10+4*i, 200-5*i, false);
      READLN;

      RestoreRegion(buffer2,0,199,false)
            {Restore background}
   END;  {for}

   READLN;

   TEXTMODE    {Return to the standard text screen}

END.
```

End Listings

SemiDisk®

has an *attractive* personality.



"A while back I got a SemiDisk to help me with my database work. A SemiDisk is like a RAM-disk only a whole lot better. It doesn't sit in my main or EMS memory, and, using the Battery Backup, it's like permanent storage.

"That SemiDisk makes light work of the jobs that were sending my hard disk to an early grave. And SemiDisk has no head to crash; no moving parts to wear out. With all the time it saves me, I figure it paid for itself in just a couple of months.

"Then I heard programs like Microsoft Windows could use my SemiDisk for temporary files instead of using EMS. So I moved them over to the SemiDisk, too. The quiet speed of it is almost elegant!

"My boss wanted to try my SemiDisk on the company LAN server, but I told him to get his own. A couple of days later, he was wearing a grin as big as mine. I guess he likes his SemiDisk too.

"One morning I booted up my computer and there was my word processor waiting for me on the SemiDisk. I swear I didn't put it there! After I tried it, I knew it was there to stay.

"Meanwhile, I've found a new use for my hard disk, too. It's great for backing up my SemiDisk!"

I/O mapped SemiDisk goes in standard PC, XT or AT expansion slot. Priced at just $495 for 512K, $795 for 2Mb. Battery Backup $130. Up to 8Mb per drive. Call or write for further information or to place an order.

SemiDisk

End the waiting.

SemiDisk Systems, Inc.
P.O. Box GG
Beaverton, OR 97075
(503) 626-3104

VISA MasterCard

CIRCLE 85 ON READER SERVICE CARD

C CODE FOR THE PC

source code, of course

C Source Code

Bluestreak Plus Communications (two ports, programmer's interface, terminal emulation)	$400
Greenleaf Data Windows (windows, menus, data entry, interactive form design)	$315
Barcode Generator (specify Code 39 (alphanumeric), Interleaved 2 of 5 (numeric), or UPC)	$300
GraphiC 4.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$275
Vitamin C (MacWindows)	$200
Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF)	$160
Greenleaf Functions (296 useful C functions, all DOS services)	$160
Essential C Utility Library (400 useful C functions)	$160
Essential Communications Library (C functions for RS-232-based communication systems)	$160
PC/IP (CMU/MIT TCP/IP implementation for PCs)	$100
B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
The Profiler (program execution profile tool)	$100
Entelekon C Function Library (screen, graphics, keyboard, string, printer, etc.)	$100
Entelekon Power Windows (menus, overlays, messages, alarms, file handling, etc.)	$100
QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	$90
CBTree (B+tree ISAM driver, multiple variable-length keys)	$80
ME (programmer's editor with C-like macro language by Magma Software)	$75
Wendin PCNX Operating System Shell	$75
Wendin PCVMS Operating System Shell	$75
Wendin Operating System Construction Kit	$75
EZ_ASM (assembly language macros bridging C and MASM)	$60
Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm modem card)	$50
Heap Expander (dynamic memory manager for expanded memory)	$50
Make (macros, all languages, built-in rules)	$50
Vector-to-Raster Conversion (stroke letters & Tektronix 4010 codes to bitmaps)	$50
Coder's Prolog (inference engine for use with C programs)	$45
PC/MPX (light-weight process manager; includes preemption and cooroutine packages)	$45
Biggerstaff's System Tools (multi-tasking window manager kit)	$40
TELE Kernel (Ken Berry's multi-tasking kernel)	$30
TELE Windows (Ken Berry's window package)	$30
Clisp (Lisp interpreter with extensive internals documentation)	$30
Translate Rules to C (YACC-like function generator for rule-based systems)	$30
6-Pack of Editors (six public domain editors for use, study & hacking)	$30
ICON (string and list processing language, Version 6 and update)	$25
LEX (lexical analyzer generator)	$25
Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	$25
C Compiler Torture Test (checks a C compiler against K & R)	$20
PKG (task-to-task protocol package)	$20
A68 (68000 cross-assembler)	$20
Small-C (C subset compiler for 8080 and 8088)	$20
tiny-c (C subsubset interpreter including the tiny-c shell)	$20
Xlisp 1.5a (Lisp interpreter including tiny-Prolog in Lisp)	$20
List-Pac (C functions for lists, stacks, and queues)	$20
XLT Macro Processor (general purpose text translator)	$20
C Tools (exception macros, wc, pp, roff, grep, printf, hash, declare, banner, Pascal-to-C)	$15

Data

DNA Sequences (GenBank 48.0 of 10,913 sequences with fast similarity search program)	$150
Protein Sequences (5,415 sequences, 1,302,966 residuals, with similarity search program)	$60
Webster's Second Dictionary (234,932 words)	$60
U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	$35
The World Digitized (100,000 longitude/latitude of world country boundaries)	$30
KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format)	$30
NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	$15
U. S. Map (15,701 points of state boundaries)	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas USA 78750-3409

CIRCLE 250 ON READER SERVICE CARD

Voice: (512) 258-0785
BBS: (512) 258-8831
Email: FidoNet 1:382/12

Free surface shipping on prepaid orders

MasterCard/VISA

68020
UNIX*COMPATIBLE MULTIUSER SYSTEM

The MPULSE Model 20

Multi-Processor Design:

Separating application processing from I/O is a well understood goal in multiuser supermicrocomputer design. As the real UNIX system bottleneck is disk I/O bandwidth, not raw processor speed, LPC has spent a great deal of time and development effort in optimizing disk I/O throughput. The result is a disk I/O subsystem unparalleled in the under $20,000 microcomputer class.

A fully asynchronous, high speed DMA channel links the MC68020 to the dual MC68000 I/O processors. A full 2 Mbytes of I/O processor memory is available for disk caching. The disk cache utilizes a least recently used, delayed write algorithm to achieve hit rates exceeding 90%.

In addition to disk caching, LPC has extended the conventional UNIX caching mechanism with a new virtual caching technique, implemented in the kernel itself. The Dynamic Kernel Cache (DKC) distributes available memory between user processes and the internal UNIX cache. This dynamic allocation technique allows a much more efficient use of main memory with cache efficiency increased to over 80%, much higher than the conventional UNIX caching mechanism.

Operating System:

The MPULSE Model 20 hosts LPC's System V operating system, derived from the industry standard UNIX System V. The MPULSE System V port is tailored to complement the MPULSE hardware while retaining compatibility at all levels with the generic UNIX System V operating system. Areas of optimization and additional features include:

- Full demand-paged virtual memory
- Kernel portions of the terminal and mass storage I/O disciplines are distributed to the appropriate I/O processors
- Berkeley enhancements
- Dynamically distributed ramdisks
- On-line Winchester disk bad block replacement
- On-line device configuration
- True multisector I/O for streaming tape operation
- Support for implementing concurrent operating systems
- General purpose user-accessible SCSI device driver
- Automatic bi-directional modem support
- MWindows windowing capability

Base System Includes:

- **16 MHz MC68020 host processor**
- **DUAL 12 MHz MC68000 I/O sub-system**
- **4 Mbytes main memory**
- **2 Mbytes of dedicated disk cache memory**
- **Demand-Paged Virtual Memory**
- **Sophisticated LRU caching algorithm**
- **Dynamic Kernel Cache (DKC)**
- **MWindows**
- **50 MByte hard disk expandable to 0.5 gigabytes**
- **800 Kbyte floppy drive**
- **60 Mbyte cassette tape backup**
- **8 serial ports expandable to 40 serial ports**
- **LPC System V derived from UNIX System V**
- **Berkeley Enhancements**
- **NCR Tower object code compatibility**

Base System Price:

$5995.00
Call (214) 340-5172

Warranty:
90 day (extended warranty available)
15 day money back evaluation period

LPC Logic Process Corporation
10355 Brockwood Road Dallas, Texas 75238

DKC and MWindows are trademarks of LPC.
UNIX is a trademark of AT&T.
Tower is a trademark of NCR.

CIRCLE 169 ON READER SERVICE CARD

Listing One *(Text begins on page 62.)*

```
/*  A C implementation Bickel's name comparison          */
/*  algorithm, CACM 30/3 (March 1987), p. 244.            */
/*  This is generic C code and should work on any          */
/*  compiler with no modification.                         */
/*  Jim Howell, March 1987.                                */

/*  This code is placed in the public domain.  You  are  */
/*  free to use it in any way you see fit.                 */


#include <ctype.h>
#include <stdio.h>


unsigned char    MaskArray [52] ={0, 0x40,          /*  a  */
                                 3, 0x80,          /*  b  */
                                 2, 0x20,          /*  c  */
                                 1, 0x10,          /*  d  */
                                 0, 0x20,          /*  e  */
                                 2, 0x10,          /*  f  */
                                 2, 0x08,          /*  g  */
                                 1, 0x08,          /*  h  */
                                 0, 0x10,          /*  i  */
                                 3, 0x08,          /*  j  */
                                 3, 0x20,          /*  k  */
                                 1, 0x04,          /*  l  */
                                 2, 0x04,          /*  m  */
                                 0, 0x08,          /*  n  */
                                 0, 0x04,          /*  o  */
                                 2, 0x02,          /*  p  */
                                 3, 0x10,          /*  q  */
                                 1, 0x02,          /*  r  */
                                 0, 0x02,          /*  s  */
                                 0, 0x01,          /*  t  */
                                 1, 0x01,          /*  u  */
                                 3, 0x40,          /*  v  */
                                 2, 0x01,          /*  w  */
                                 3, 0x04,          /*  x  */
                                 3, 0x02,          /*  y  */
                                 3, 0x01};         /*  z  */


/*  Demonstrate the algorithm with some short examples.  */
main ()
{
    unsigned char    LetterSet0 [4],
                     LetterSet1 [4];

    MakeLetterSet ("ecdysiast", LetterSet0);
    MakeLetterSet ("ecstasy", LetterSet1);
    printf ("The likeness of 'ecdysiast' and 'ecstacsy' is %d.\n",
      CompareSets (LetterSet0, LetterSet1));

    MakeLetterSet ("ectoplasm", LetterSet1);
    printf ("The likeness of 'ecdysiast' and 'ectoplasm' is %d.\n",
      CompareSets (LetterSet0, LetterSet1));

    MakeLetterSet ("edcysiast", LetterSet1);
    printf ("The likeness of 'ecdysiast' and edcysiast' is %d.\n",
      CompareSets (LetterSet0, LetterSet1));
    }    /*  main  */


/*  Make the letter set by going through the string       */
/*  one character at a time.                               */

MakeLetterSet (Name, Mask)
    char            *Name;
    unsigned char   *Mask;
{
    char            *pC;
    unsigned char   *pM;

int            I,
               Lk;

pC =Name;
for (I =0; I <4; ++ I)
    Mask [I] =0;
while (*pC)        {
    /*  Use letters only, and convert    */
    /*  uper to lower case.              */
```

```
        if (isalpha (*pC))      {
            pM =&MaskArray [2 *(tolower (*pC) -'a')];
            Mask [*pM] |=*(pM +1);
            }
        ++ pC;
        }
    }    /*  MakeLetterSet  */


/*  This particular version constructs a mask by using     */
/*  a logcal AND of the relevant bytes of the two sets.    */
/*  The rightmost bit is checked to see if the likeness    */
/*  score needs to be increased by the appropriate         */
/*  weight, and the mask is shifted right one bit.  An     */
/*  alternative would be to use a bit mask and to          */
/*  shift it instead of the name mask.  The two bytes      */
/*  of the least common letters are checked for content    */
/*  to eliminate any unnecessary calculations.             */

CompareSets (Set1, Set2)
    unsigned char    *Set1,
                     *Set2;
{
    unsigned char    Mask;
    int              I,
                     Lk;

    Lk =0;
    /*  For the first byte.  */
    Mask =Set1 [0] & Set2 [0];
    for (I =0; I <7; ++ I)      {
        if (Mask & 0x01)
            Lk +=3;
        Mask >>=1;
        }

    /*  The second byte.  */
    Mask =Set1 [1] & Set2 [1];
    for (I =0; I <5; ++ I)      {
        if (Mask & 0x01)
            Lk +=4;
        Mask >>=1;
        }

    /*  The third byte.  */
    Mask =Set1 [2] & Set2 [2];
    if (Mask)
        for (I =0; I <6; ++ I)      {
            if (Mask & 0x01)
                Lk +=5;
            Mask >>=1;
            }

    /*  The last byte is more complicated.  */
    Mask =Set1 [3] & Set2 [3];
    if (!Mask)
        return (Lk);
    if (Mask & 0x01)
        Lk +=9;                     /*  z  */
    Mask >>=1;
    if (Mask & 0x01)
        Lk +=8;                     /*  y  */
    Mask >>=1;

    if (Mask & 0x01)
        Lk +=8;                     /*  x  */
    Mask >>=1;
    if (Mask & 0x01)
        Lk +=8;                     /*  j  */
    Mask >>=1;
    if (Mask & 0x01)
        Lk +=7;                     /*  q  */
    Mask >>=1;
    if (Mask & 0x01)
        Lk +=7;                     /*  k  */
    Mask >>=1;
    if (Mask & 0x01)
        Lk +=6;                     /*  v  */
    Mask >>=1;
    if (Mask & 0x01)
        Lk +=6;                     /*  b  */
    Mask >>=1;

    return (Lk);
    }    /*  CompareMasks  */
```

End Listings

Three in One

Finally, one program that does all you have been asking for in a screen generator.

ScreenCODE--It's easy, it's fast, and it writes screen code in all the popular languages--BASIC, TurboPascal, dBase II, III, III+, and C. You can build boxes, pull down menus, input screens -- everything you need, and you spend your time creating, not coding.

ScreenSNAP--Capture screens from ANY PROGRAM and use ScreenCODE to incorporate it into your program. Use captured screens as is, or modify them to suit you. Now you won't have to reinvent the wheel when you see a screen you like.

SlideSHOW--No screen generator should be without this feature. Easier to use than Bricklin, you can build prototypes and self running demonstrations. Nothing will sell your programming skills better.

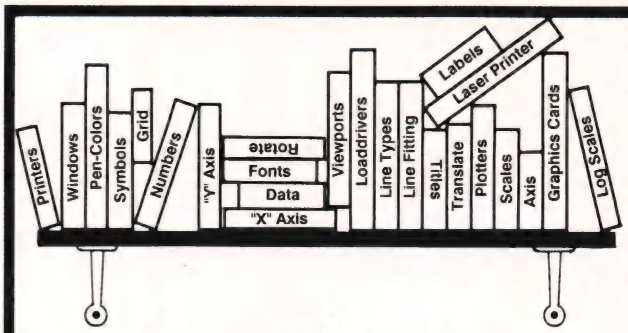
You get all three in one package with 90 day free support and our **30 DAY MONEY BACK GUARANTEE**.

Order before December 31, 1987 and get 75 of our most popular screen templates FREE - ready to use as is or modify as needed.

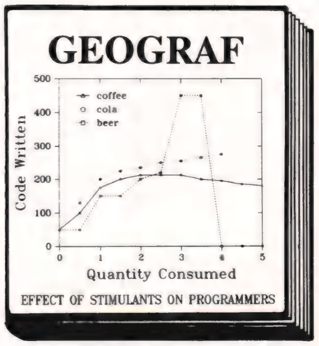
Now through the end of the year get all three for **$149.00**. (Shipping and handling included.)

To order call 1(800)223-3419.

Accept Visa, MasterCard, Check, Money Order, C.O.D.
Centurion Systems 2724 W. Waco Drive, Waco, TX 76710
CIRCLE 236 ON READER SERVICE CARD



GRAPHICS LIBRARY For PC programmers



Create custom graphs of any type with GEOGRAF's powerful library of subroutines. Plot on screen, dot matrix printer, laser printer and pen plotter at maximum device resolution without changing your code. FORTRAN version is fully compatible with mainframe Calcomp graphics library. Available for FORTRAN and BASIC for all PC compatibles.

MONEY BACK GUARANTEE

800-822-2669

GEOCOMP Corp.
342 Sudbury Road
Concord, MA 01742

CIRCLE 111 ON READER SERVICE CARD

Listing One *(Text begins on page 54.)*

```
        PUBLIC Lhandl_
        EXTERN c_ds,Hhandl_      ;See text for NEAR/FAR choice.
Lhandl_:
        pushf                    ;Push all registers. The order is
        push    ds               ;  arbitrary so long as it is known to the
        push    es               ;  interrupt function in Hhandl().
        push    bp
        push    si
        push    di
        push    dx
        push    cx
        push    bx
        push    ax
        mov     ds,cs:c_ds       ;Adjust DS and whatever other registers
                                 ;  your version of C requires.
        call    Hhandl_          ;Call high-level handler.
        pop     ax               ;Restore registers.  These may have been
        pop     bx               ;  altered by the interrupt function
        pop     cx               ;  called by the high-level handler.
        pop     dx
        pop     di
        pop     si
        pop     bp
        pop     es
        pop     ds
        popf
        ret       02             ;Substitute db 0cah,2,0 if a NEAR procedure
```

End Listing One

Listing Two

```
Hhandl(fake)
   int fake;
{
  REGS *regs = &fake;
  char serv;

  serv=regs->ax>>8;             /* preserve service # from caller's AH */
  interrupt(NEW16,regs);        /* chain to keyboard i/o bios routine */
  if(serv==0) switch(regs->ax){ /* if get-input service, test return */
     case HOTKEY1: hot1();       /* run through the hot keys */
       break;
     case HOTKEY2: hot2();
        break;
     case HOTKEY3: hot3()
        break;
     ...and so forth...
     }
}
```

End Listing Two

Listing Three

```
;interrupt(nn,pptr)
; int nn;
; REGS *pptr;
  PUBLIC interrupt_
interrupt_:
  push  bp
  mov   bp,sp                ;Set up base pointer for stack
  push  ds                   ;Push whatever registers need to be preserved
  mov   al,0cdh              ;CD (do-interrupt) machine code into low byte
  mov   ah,[bp+6]            ;Now reverse-byte AX reads "CDnn": nn=int#
  mov   cs:intnum[0],ax      ;Move interrupt code to instruction site
  lds   bx,[bp+8]            ;Load seg:ofs of struct pointer into DS:BX
  mov   ax,[bx+0]            ;Move structure's AX value into AX
  push  [bx+2]               ;Push final BX value onto stack
  mov   cx,[bx+4]            ;Load the rest, skipping BP & FLAGS
  mov   dx,[bx+6]
  mov   di,[bx+8]
  mov   si,[bx+10]
  mov   es,[bx+14]
  mov   ds,[bx+16]           ;Pointer gone: must pop BX
  pop   bx
  push  bp                   ;Somebody else may have stolen things beforehand
```

```
intnum: dw    00              ;Scene of crime of self-modifying code
  pop   bp
  push  bx                    ;Save returned DS & BX so we can use pointer
  push  ds
  lds   bx,[bp+8]             ;Load pointer
  mov   [bx+0],ax             ;Inverse of sequence above
  mov   [bx+4],cx
  mov   [bx+4],cx
  mov   [bx+6],dx
  mov   [bx+8],di
  mov   [bx+10],si
  mov   [bx+14],es
  pushf                       ;Must consider flags this time
  pop   [bx+18]               ;Flags into flags slot
  pop   [bx+16]               ;DS into structure
  pop   [bx+2]                ;And BX
  pop   ds                    ;Recover preserved registers
  pop   bp
  ret
```

End Listing Three

Listing Four

```
;chgint(old,new,lowhandler)
; int old,new,(*lowhandler)(); /*old vector, new, pointer to handler*/
   PUBLIC chgint_
chgint_:
  push bp
  mov  bp,sp                ;Set up base pointer to stack
  push ds
  mov ah,35h                ;First check to see if new vector is in use
  mov al,[bp+8]             ;Destination interrupt # into AL
  int 21h                   ;Present vector contents into ES:BX
  mov ax,es                 ;Set up for possible error return in AX
  or  ax,bx                 ;Are both seg & offset zero?
  jnz error                 ;if so, report error and exit
  mov ah,35h                ;Get-vector service again
```

(continued on next page)

Listing Four *(Listing continued, text begins on 54.)*

```
    mov al,[bp+6]          ;Old interrupt # into AL
    int 21h                ;Pick up "old" in ES:BX
    mov ax,es              ;ES:BX->DS:DX for function 25h call
    mov ds,ax
    mov dx,bx
    mov ah,25h
    mov al,[bp+8]          ;Interrupt destination into AL
    int 21h                ;Load DS:DX into table at "new"
    lds dx,[bp+10]         ;Low load pointer to low-level handler
    mov ah,25h
    mov al,[bp+6]          ;Old int# into AL again
    int 21h                ;Put function pointer into table
    pop ds                 ;Restore registers
    pop bp
    mov ax,0               ;return of zero in AX means O.K.
error:  ret
```

End Listing Four

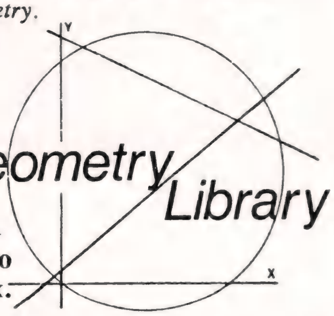
Listing Five

```
#include <regsdef.h>          /* typedef of REGS structure */
#define OLDINT  XX            /* interrupt captured */
#define NEWINT  YY            /* new location for old vector */
/******************************/
Hhandl(xx)
 int xx;
{
 REGS *regs = &xx;
/*
*  Do whatever it is here that needs to be done. Obviously, other
*  C functions could be called from this one. Don't forget to chain
*  the interrupt so that other resident programs get their chances.
*  If a hardware interrupt is being captured and your handler is fairly
```

Introductry Offer $99.95

The added plus you need for developing sophisticated computer graphics, CAD, and programs that use computational geometry.

You save time and money with its flexibility.



TurboGeometry Library

An excellent addition to **Borland's Turbo Graphix Toolbox.**

Over 150 ready to use routines, such as Equations of Lines, Circles, Arcs & Planes • Intersection of Lines, Circles, Planes • Curves • 2&3 Dimensional Transforms • Polygons • Hidden Lines • Volumes • Areas • Prespectives • Polygon Clipping and more....

Manual, full source code and sample programs.All for the introductry price of $99.95US. Add $5.00 for SH. Tex Res add 8% ST. 30 day money back guarentee. VISA,MC,Check MO accepted. Needs Turbo Pascal 2.0+, IBM PC(Comp),Zenith Z100,MS/PC DOS 2.0+.

DISK SOFTWARE, INC. , 2116 E Arapaho, # 487
Richardson, Texas 75081 (214) 423-7288

CIRCLE 165 ON READER SERVICE CARD

Easy to C

C is a great programming language. Now the C *WORKSHOP* makes it easy.

Interactive software teaches you C with immediate feedback on your program exercises.

The C *WORKSHOP* has everything you need to learn C and write your own programs, too. You get a fast editor, standard C compiler, and online help.

Let the other guy struggle with confusing books and compilers. Join AT&T and other major companies now using the *C WORKSHOP*. Columnist Adam Green calls it "the most intriguing new type of training system I've ever seen." (InfoWorld 1/27/86)

Wordcraft

Quality software since 1981

Order Information

To order the *C Workshop*, call toll-free (800) 227-2400 ext. 955 day or night (Visa, MC, or AmEx). Or send check to Wordcraft, 3827 Penniman Ave., Oakland, CA 94619. $69.95 plus $5.00 shipping (Priority Mail). In CA, add $4.90 sales tax.

CIRCLE 163 ON READER SERVICE CARD

```
*  lengthy, you may need to set a static flag to block reentry and
*  include _inb(0x20,0x20)--an EOI to the PIC.
*/
}
/****************************/
main()          /* Initialization routine */
{
  int Lhandl();  /* tell the compiler about the low-level handler */
  REGS rr;       /* allocate space for initializers's interrupt calls */

  saveds();                    /* save data segment */
  /*
  *  Set up the high-level handler here by initializing globals--e.g.,
  *  screen parameters, addresses in the 00:400h memory area, strings for
  *  hot-key substitutions, a pointer to the DOS-can-be-interrupted flag,
  *  a pointer to the  memory allocation chain, etc. Do it now, in C,
  *  while you are not in residence.
  */
  if(chgint(OLDINT,NEWINT,Lhandl))     /* test & swap vectors */
    puts("\n\nVECTOR IN USE\7\7");
  else{
     rr.ax = 0x3100;                   /* exit, stay resident service. */
     rr.dx = PROGRAMSIZE;              /* PROGRAMSIZE in paragraphs */
     puts("\n\nDONE & INSTALLED");
     interrupt(0x21,&rr);
     }
}
```

End Listings

C Programmers: Combine C and COMMON LISP to Increase the Power of Your Software

TransLISP PLUS™

Simple.

Add LISP features to your software without making it a full time job. The TransLISP PLUS tutorial, on-line help, and 30 sample programs with commented source make it easy.

Practical.

Start by modifying the LISP sample programs and including them in a system you wrote in C. Yes, in C! TransLISP PLUS includes a C Language Interface that lets you integrate your Microsoft C code and libraries with all or portions of our LISP interpreter.

Use TransLISP PLUS to add natural or command language features to replace menus . . . or to flexibly manage related but disparate information. Code from C libraries provided by other vendors can be integrated into your program to perform tasks not normally part of LISP.

Thorough.

TransLISP PLUS took over 400 primitives from the most widely used and respected LISP standard, COMMON LISP, and made it available on IBM PCs, XTs, ATs, and virtually every other MSDOS machine. So now you can work with anything from a $700 PC to a $7000 PC.

The utilities toolbox is included at no charge with a built-in editor, pretty printer, cross reference, and additional debugging tools.

An optional Runtime encrypts your source code so that you can distribute your applications safely. You pay no royalties.

Requires MSDOS 2.0+, 320K RAM, and a 360K floppy.

MONEYBACK GUARANTEE

Try TransLISP PLUS ($195) for 30 days — if not satisfied get a full product refund. The Optional Runtime is available for $150. Or start by learning LISP with TransLISP ($95) then upgrade to PLUS for $158.

Call (800) 255-4659
In MA (617) 331-0800

The Coder's Source™ 541-D Main St., Suite 412, So. Weymouth, MA 02190 **CIRCLE 315 ON READER SERVICE CARD**

Listing One *(Listing continued, text in October.)*

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * ****
;_______________________________________________________________________
;
; MDoWarn -- Control call to put up a warning
;
; Entry:    A0 -> IOQelement
;_______________________________________________________________________

MDoWarn     move    CSParam(a0),D0      ; get the error code into D0
            bsr.s   DoWarn              ; warn 'em and return status in D0
            bra     AbusExit            ; and exit

; Put up alerts
;________________________________________________________________
;
;   DoWarn -- Warn the user...  Give a beep, and display a dialog;
;             wait for their choice, then try the NNNN one more time.
;             If they choose "Use new address" from Mismatch dialog,
;             set SysLAPaddr and SysNetNum to zero before exiting.
;
;   Entry:   D0 = PortNotCf
;                 noAnswer
;   Exit:    D0 = -4001 (user clicked OK/Try again)
;   -4002  (user clicked Use New)
;  -193  (ResFNotFound)
;________________________________________________________________

resfile     EQU     -2                  ; res file number
dlgwindow   EQU     -6                  ; dialog window handle
warning     EQU     -8                  ; error number/return status
MyCurMap    EQU     -10                 ; save the current res file

DoWarn      _SUBR   10                  ; Warn the user about troubles

            move.w  D0,warning(A6)      ; remember the warning
            move.w  CurMap,MyCurMap(A6) ; and the current res file
            InitCursor                  ; make it an arrow again

; Open our resource file.

            subq.l  #2,sp               ; make space for result
            pea     fileName            ; point to file name
            OpenResFile
            move.w  (SP)+,resfile(A6)   ; save the resfile number
            cmp.w   #-1,resfile(a6)     ; check for failure
            bne.s   @3                  ; branch if ok
            move.w  #60,-(A7)           ; else beep (long)
            SysBeep
            move.w  #ResFNotFound,D0    ; return bad status
            bra.s   @20                 ; and quit

; beep at 'em
@3
            move.w  #6,-(A7)            ; 1/10 second beep
            _SysBeep

; choose a dialog to display
            move.w  #PortNCalrt,D0
            cmp.w   #PortNotCf,warning(a6) ; which warning?
            beq.s   @5
            move.w  #Noansalrt,D0       ; noAnswer dialog

; now display the dialog
@5
            subq.l  #4,sp               ; space for result of _GetNewDialog
            move.w  D0,-(sp)            ;    dialog resource ID
            clr.l   -(sp)               ;    dialog record in heap
            move.l  #-1,-(sp)           ;    in front of other windows
            _GetNewDialog
            move.l  (SP)+,dlgwindow(A6) ; save the dialog's handle

;  Now do the dialog stuff

            subq.l  #2,sp               ; result on stack
            clr.l   -(sp)               ; normal filterproc
            pea     4(sp)               ; point to result space
            _ModalDialog                ;    Do it

; discard dialog
```

```
          move.l  dlgWindow(a6),-(sp) ; point to dialog
          _DisposDialog

; What did they hit?

          move.w  (sp)+,d0            ; get the button's item #
          cmp.w   #1,D0               ; (Try Again or OK (=1)) or Use New?
          beq.s   @10                 ; go if not "Use New"
          clr.b   SysLAPAddr(a2)      ; otherwise, Use New
          clr.w   SysNetNum(a2)       ;   and reset net and node adrs

@10       neg.l   D0                  ; item will be 1 or 2; return -4001
          sub.l   #4000,D0            ;      or -4002 as the status
          move.w  D0,warning(A6)      ; save it

; discard resource file

          move.w  resfile(A6),D0      ; get the refnum
          cmp.w   MyCurMap(A6),D0     ; was it the current resource file
          beq.s   @15                 ; go if so (someone else opened it)
          move.w  D0,-(sp)            ; else, push it
          CloseResFile                ; and close it
@15       move.w  warning(A6),D0      ; get the status from the NNNN

@20  ; D0 is result code for this routine
         _SUBEND 'DOWARN  '           ; and exit

FileName  DC.B    15
          DC.B    'Async AppleTalk'
          DC.B    'V1.2a6'
          ALIGN   2

; ****** end of lap.a
```

End Listing One

Listing Two

```
CRC Calculations

     This file contains a CRC calculation in Pascal.  It was used with
     preliminary versions of Async AppleTalk, and computes the same
     function as the code in the M68000 listing.

     The NextCRC algorithm simulates the feedback shift register which
     normally implements a CRC calculation.  NextCRC takes each four-
     bit nibble of the input char and uses a table (crctbl) to select
     a mask which is exclusive-or'd with the current CRC accumulator.
}

{ pseudo-CONST -- put this in the initialization code of your program

crctbl[00] := $0000; crctbl[01] := $CC01;
crctbl[02] := $D801; crctbl[03] := $1400;
crctbl[04] := $F001; crctbl[05] := $3C00;
crctbl[06] := $2800; crctbl[07] := $E401;
crctbl[08] := $A001; crctbl[09] := $6C00;
crctbl[10] := $7800; crctbl[11] := $B401;
crctbl[12] := $5000; crctbl[13] := $9C01;
crctbl[14] := $8801; crctbl[15] := $4400;
}

VAR crctbl : array [0..15] of integer;

function NextCRC (crc : integer; c : QDbyte) : integer;

VAR
     j : integer;

BEGIN
     j := crctbl[ band(bxor(crc,c),$000F) ];
     crc := bxor(bsr(crc,4),j);
     c := bsr(c,4);
     j := crctbl[ band(bxor(crc,c),$000F) ];
     crc := bxor(bsr(crc,4),j);
     nextcrc := crc;
END; { NextCRC }

function crc16 (p : qdptr; len : integer) : integer;
```

(continued on next page)

function libraries
disassemblers
compilers
text editors
text filters
communications support
text formatters
interpreters
bulletin boards
co-routines
compiler compilers
window packages
assemblers
games
tutorials
math packages
link editors
languages
cross compilers
pre-processors
function libraries
disassemblers
compilers
text editors

The C Users' Group Library

A Directory of Public Domain C Source Code

Send $10 for Directory. Write or call for more details on over 100 volumes of Public Domain C Source Code.

The C Users' Group
PO Box 97
McPherson, KS 67460
(316) 241-1065

CIRCLE 181 ON READER SERVICE CARD

ICs PROMPT DELIVERY!!!
SAME DAY SHIPPING (USUALLY)
QUANTITY ONE PRICES SHOWN **for SEPT. 20, 1987**

OUTSIDE OKLAHOMA: NO SALES TAX

DYNAMIC RAM			
1Mbit	1000Kx1	100 ns	$27.50
1Mbit	256Kx4	120 ns	32.00
51258	*256Kx1	100 ns	6.50
4464	64Kx4	150 ns	3.60
41256	256Kx1	80 ns	5.50
41256	256Kx1	100 ns	4.85
41256	256Kx1	120 ns	3.95
41256	256Kx1	150 ns	3.50
41264	2-PORT	120 ns	5.25
EPROM			
27C512	64Kx8	200 ns	$15.50
27C256	32Kx8	250 ns	6.75
27256	32Kx8	250 ns	5.60
27128	16Kx8	250 ns	5.35
STATIC RAM			
43256L-12	32Kx8	120 ns	$11.75
5565PL-15	8Kx8	150 ns	3.25

640K MOTHERBD UPGRADE: Zenith 150, IBM PC/XT, Compaq Portable & Plus; hp Vectra

*STATIC COLUMN DRAM FOR 386 COMPAQ

80387-16	$540.00
80287-8	$250.00
8087-2	$160.00

OPEN 6½ DAYS, 7:30 AM-10 PM: SHIP VIA FED-EX ON SAT.

SUNDAYS & HOLIDAYS: SHIPMENT OR DELIVERY, VIA U.S. EXPRESS MAIL

SAT DELIVERY INCLUDED ON FED-EX ORDERS RECEIVED BY:
Th: Std Air $4/1 lb
Fr: P-1 $10.50/2 lbs

MasterCard/VISA or UPS CASH COD
Factory New, Prime Parts μPoo
MICROPROCESSORS UNLIMITED, INC.
24,000 S. Peoria Ave.,
BEGGS, OK. 74421 **(918) 267-4961**
No minimum order. Please note that prices are subject to change. Shipping & insurance extra, & up to $1 for packing materials. **Orders received by 9 PM CST can usually be delivered the next morning, via Federal Express Standard Air @ $4.00, or guaranteed next day Priority One @ $10.50!** All parts guaranteed.

CIRCLE 105 ON READER SERVICE CARD

In conjunction with Glockenspiel of Dublin

C++ TRAINING

*Programming Workshop
November 16-20
December 14-18
January 11-15

*Overview for Developers
November 26
December 9
January 20

On site courses available.
For more information and reservation:

Semaphore, Inc.
16 Haverhill Street, Andover, MA 01810
(617)685-1478

CIRCLE 308 ON READER SERVICE CARD

Listing Two *(Listing continued)*

```
VAR
   i,j : integer;      { sixteen bits wide }
   c   : qdbyte;       { an eight bit value }
   crc : integer;      { the CRC accumulator }

BEGIN
   crc := 0;
   for i := 1 to len do begin
       c := p^;
       p := pointer(ord(p) + 1);
       crc := NextCRC(crc,c);
   end;
   crc16 := crc;
END; { crc16 }
```

End Listing Two

Listing Three

```
;
; _AssumeEq  Arg1, Arg2 -- macro to generate a compile-time error if two
;                          arguments are unequal.
;
;         To optimize code size, we will be making various assumptions,
;         mainly as to offset values. This macro is a way of formalizing
;         those assumptions within the code.
;

     BLANKS ON
     STRING ASIS

          MACRO
          _AssumeEq
               IF  &Eval(&Syslst[1]) <> &Eval(&Syslst[2]) THEN
                   _ERR            ; Invalid statement - will cause error
               ENDIF
          ENDM
;
```

Query+MANAGER=
Q-MAN

Q-MAN, a fast true relational data base management system which supports interactive and imbedded queries, library routines, forms, and will be multi-user or distributed. Uses QUEL and SQL languages, B+ trees, and sort-merge joins. Free multi-user upgrade when released first quarter of 1988.

Runs on:	**single-user**
SYS 5	$1395
4.2	1395
MS-DOS™	329

To order
Call collect
(608) 271-2171

Breakpoint Computer Systems, Inc.
6701 Seybold Road, Suite 204
Madison, WI 53719

Include machine name and operating system when ordering.

Visa and Mastercard accepted.
MS-DOS™ is a registered trademark of Microsoft Corporation.

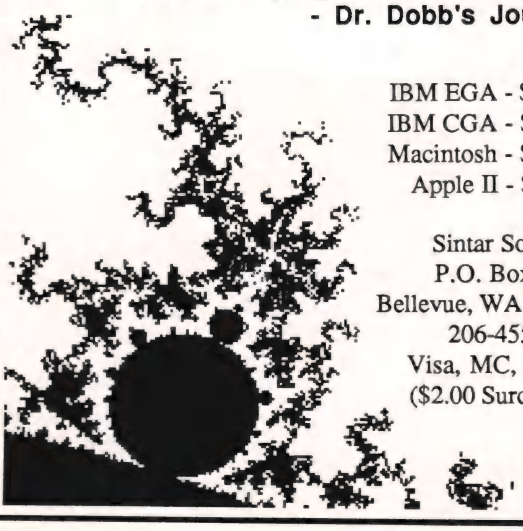
CIRCLE 327 ON READER SERVICE CARD

Explore the beauty of fractals!

FractalMagic™ lets you investigate these beautiful images on your personal computer.

"I can't imagine a better way to waste an afternoon or two. No color PCompatible is complete without a copy of this; get one and see what I mean." Jerry Pournelle - BYTE Magazine

"A slick, fast program." Ray Duncan - Dr. Dobb's Journal



IBM EGA - $25.00
IBM CGA - $25.00
Macintosh - $25.00
Apple II - $25.00

Sintar Software
P.O. Box 3746
Bellevue, WA 98009
206-455-4130
Visa, MC, AmEx
($2.00 Surcharge)

```
; _StatCount Arg1 -- increment a statistics count if stat keeping is enabled
;
;          Assumes A2 points to the driver variables
;

           MACRO
           _StatCount
                IF  debug THEN
                     ADDQ.L  #1,&Syslst[1](A2); Update the count
                ELSE
.*                   nop   ; commented out
                ENDIF
           ENDM

;
; _Subr         -- assembles a "Link A6,#???"
;                  works for _SUBR <no param>   and   _SUBR ###
;
           MACRO
           _Subr     &size
           IF        &size = '' THEN
           Link      A6,#0
           ELSE
           Link      A6,#(-&size)
           ENDIF
           ENDM

;
; _Subend  NAME,$xx  -- Subroutine epilog
;                          If debugging, put in Unlk and the name
;
           MACRO
           _Subend &name
           Unlk      A6         ; unlink the stack frame
           rts                  ; and return
           DC.B      &name      ; the name
           ALIGN     2
           ENDM
```

End Listings

ADD TO THE POWER OF YOUR PROGRAMS WHILE YOU SAVE TIME AND MONEY!

CBTREE does it all! Your best value in a B+tree source!

Save programming time and effort.
You can develop exciting file access programs quickly and easily because CBTREE provides a simple but powerful program interface to all B+tree operations. Every aspect of CBTREE is covered thoroughly in the 70 page Users Manual with complete examples. Sample programs are provided on disk.

Gain flexibility in designing your applications.
CBTREE lets you use multiple keys, variable key lengths, concatenated keys, and any data record size and record length. You can customize the B+tree parameters using utilities provided.

Your programs will be using the most efficient searching techniques.
B+trees use efficient search techniques that require fewer disk seeks than other methods. CBTREE guarantees an optimized maximum search path and always remains balanced. CBTREE is optimized for speed. You will be using a commercial quality, reliable and powerful tool. CBTREE is a full function implementation of the industry standard B+tree access method and is proven in applications since 1984.

Access any record or group of records by:
- its absolute position in the index (GETFRST and GETLAST),
- its relative location in the index (GETPRV, GETNXT and GETSEQ),
- an exact match to a key (GETREC),
- a partial match to a key (GETPAR, GETALL and GETKEYS)
- a lexical relation to a key (GETLT, GETLE, GETGT and GETGE).
- You may also add, delete and update any record without the need to reorganize the index (INSERT, ISRTKY, DELETE, DELTKY and NEWLOC).
- Block retrieval calls speed up sequential processing.

Increase your implementation productivity.
CBTREE is over 6,000 lines of tightly written, commented C source code. The driver module is only 20K and links into your programs.

Port your applications to other machine environments.
The C source code that you receive can be compiled on all popular C compilers for the IBM PC and also under Unix, Xenix, and AmigaDos! No royalties on your applications that use CBTREE. CBTREE supports multi-user and network applications.

CBTREE IS TROUBLE-FREE, BUT IF YOU NEED HELP WE PROVIDE FREE PHONE SUPPORT.
ONE CALL GETS YOU THE ANSWER TO ANY QUESTION!
CBTREE compares favorably with other software selling at 2,3 and 4 times our price.
Sold on unconditional money-back guarantee.
YOU PAY ONLY $99.00 - A MONEY-SAVING PRICE!
TO ORDER OR FOR ADDITIONAL INFORMATION
CALL (703) 356-7029 or (703) 847-1743
OR WRITE

PEACOCK SYSTEMS, INC

Peacock Systems, Inc., 2108-C Gallows Road, Vienna, VA 22180

CIRCLE 316 ON READER SERVICE CARD

Listing One

(Listing continued, text begins on page 116.)

```
              Listing 1 -- calendar.c, Printed 12/31/1987

#include <stdio.h>
#include <time.h>
#include <ctype.h>
/*------------------------------------------------------------
 * CALENDAR.C   This program searches a file called "calendar"
 *              in the current directory and prints all lines
 * that start with today's or tomorrow's date. Note that this
 * behaviour differs from Unix, which allows the date to be
 * anywhere on the line. Leading whitespace on the line is
 * ignored, however. Like Unix, "tomorrow" is extended to
 * Monday if calandar is run on a weekend.
 *
 * If the date is a weekday (monday, tuesday, wednesday),
 * which can be abbreviated by the first few character of the
 * name (mon, tu, wed) then the line is printed on that day,
 * reguardless of the actual date. For example, lines starting
 * with "tue" will be printed every tuesday. Explicit dates
 * can be listed too and most common abreviations work:
 *
 *              Sept. 7, 1987
 *              September 7 '87
 *              sept. 7
 *              Sep 7 87
 *              9-7
 *              9/7
 *              9/7/87
 *              9 7 1987
 *              9-7-1987
 *
 * and so forth. Like unix, "7 September" doesn't work, however.
 * The day must follow the month.
 *
 * If a * is found in the month's section of any of the above
 * dates (as in "*-7-87" or "* 7)" then that day in any month
 * will match. If no day is specified, then all days in the
 * indicated month will match.
 *
 * Abbreviations are formed by looking at the first few
 * characters of the word. The shortest possible abbreviations
 * are:
 *
 *              su      sunday          mo      monday
 *              tu      tuesday         w       wednesday
 *              th      thursday        fr      friday
 *              sa      saturday        ja      january
 *              fe      february        mar     march
 *              ap      april           may     may
 *              jun     june            jul     july
 *              au      august          se      september
 *              o       october         n       novemeber
 *              d       december
 *
 * BUGS: * If you specify both the day of the week and the date,
 *         as in "Tuesday, July 7, 1987", the "Tuesday" is
 *         processed and the date is ignored.
 *
 *       * The following won't work:
 *                      Jun 1   49er's today
 *        because calendar will pick off 49 as the current year.
 *        Put some nonnumeric character that can't be part of a
 *        date string in front of the "49." You can use any
 *        character but a number or one of: ( - . / ' ).
 */

#define JAN     31      /* Days in the month    */
#define FEB     28
#define MAR     31
#define APR     30
#define MAY     31
#define JUN     30
#define JUL     31
#define AUG     31
#define SEP     30
#define OCT     31
#define NOV     30
#define DEC     31

int offset_to_first_of_month[] =
{
        0,                              /* Offset to the first of */
        JAN,                               /* of the month from  */
        JAN +FEB,                             /* January, 1st.  */
        JAN +FEB +MAR,
        JAN +FEB +MAR +APR,
        JAN +FEB +MAR +APR +MAY,
        JAN +FEB +MAR +APR +MAY +JUN,
        JAN +FEB +MAR +APR +MAY +JUN +JUL,
        JAN +FEB +MAR +APR +MAY +JUN +JUL +AUG,
        JAN +FEB +MAR +APR +MAY +JUN +JUL +AUG +SEP,
        JAN +FEB +MAR +APR +MAY +JUN +JUL +AUG +SEP +OCT,
        JAN +FEB +MAR +APR +MAY +JUN +JUL +AUG +SEP +OCT +NOV
};

/*-------------------------------------------------------------*/

char    *skipstuff(p)
char    *p;
{
    /* All of the following can be used in a date string */

    while(isspace(*p)||*p=='-'||*p=='/'||*p=='.'||*p==','
                                                ||*p=='\'')
        ++p;

    return p;
}

/*-------------------------------------------------------------*/

main()
{
    char        buf[132], *p;
    FILE        *fd;
    struct tm   *t;
    long        thetime;
    int         month, day, year, wday;
    int         c1, c2, c3;

    if( !(fd = fopen("calendar", "r") ) )
    {
        perror( "calandar" );
        exit(1);
    }

    thetime = time(NULL) ;
    t = localtime( &thetime );
    printf("Today is %s\n", asctime( t ) );

    while( fgets( buf, 132, fd ) )
    {
       wday  = -1;
       month = -1;
       day   =  0;
       year  =  0;
       p     = buf;

       while( isspace(*p) )
               ++p;

       if( isalpha(*p) )
       {
           c1 = tolower( p[0] );
           c2 = tolower( p[1] );
           c3 = tolower( p[2] );

           if      ( c1=='a' && c2=='p'              ) month = 3 ;
           else if( c1=='a' && c2=='u'              ) month = 7 ;
           else if( c1=='d'                         ) month = 11;
           else if( c1=='f' && c2=='e'              ) month = 1 ;
           else if( c1=='f' && c2=='r'              ) wday  = 5 ;
           else if( c1=='j' && c2=='a'              ) month = 0 ;
           else if( c1=='j' && c2=='u' && c3=='l' ) month = 6 ;
           else if( c1=='j' && c2=='u' && c3=='n' ) month = 5 ;
           else if( c1=='m' && c2=='a' && c3=='r' ) month = 2 ;
           else if( c1=='m' && c2=='a' && c3=='y' ) month = 4 ;
           else if( c1=='m' && c2=='o'              ) wday  = 1 ;
           else if( c1=='n'                         ) month = 10;
           else if( c1=='o'                         ) month = 9 ;
           else if( c1=='s' && c2=='a'              ) wday  = 6 ;
           else if( c1=='s' && c2=='e'              ) month = 8 ;
           else if( c1=='s' && c2=='u'              ) wday  = 0 ;
           else if( c1=='t' && c2=='h'              ) wday  = 4 ;
           else if( c1=='t' && c2=='u'              ) wday  = 2 ;
           else if( c1=='w'                         ) wday  = 3 ;

           if( wday >0 )
           {
               if( t->tm_wday == wday || (t->tm_wday +1) == wday)
                   fputs( buf, stdout );

               continue;
           }

           while( isalpha( *p ) || *p == '.' )
                   p++;

           p = skipstuff(p);
       }

       if( isdigit( *p ) || *p == '*' )
       {
           if( *p == '*' )
           {
               month = t->tm_mon;
               ++p;
           }
           else if( month < 0 )
               month = stoi( &p ) - 1;

           p   = skipstuff(  p );
           day = stoi      ( &p );
           p   = skipstuff(  p );

           if( (year = stoi(&p)) > 99 )
                   year -= 1900;
       }

       if( !year      )   year  = t->tm_year ;
       if( month < 0 )   month = t->tm_mon  ;
       if( !day       )   day   = t->tm_mday ;

       /* Convert the specified date to an offset, in days,
        * from Jan 1. Unix provides the offset for today in the
        * tm_yday field. It makes life difficult by making 1 Jan.
        * be 0, however. We also have to compensate for leap
        * year. If the year is a multiple of 4, it's a leap year
        * unless it's also the first year of the century (1900
        * was not a leap year, even though it's a multiple of 4).
        */

       day += offset_to_first_of_month[ month ] - 1;
       if( year % 4 == 0  &&  year % 100 != 0  && month >= 2 )
               day++;

```

(continued on page 110)

It's good for your system! **Vitamin C**

"If you need source code, make sure your wallet is wide open or get VITAMIN C.

Picking the best value package is hard... If you're a source code fanatic like me, VITAMIN C is preferable."

- Computer Language, June, 1987

Fast, flexible, versatile, reliable. Vitamin C delivers the vital combination software professionals demand to produce superior applications in dramatically less time. Highly efficient, professionally crafted C code provides lightning fast displays required by today's window intensive programs.

High level functions provide maximum productivity and require little supporting code. Extended versions of these routines add flexible control over specific details when necessary. Plus, Vitamin C's versatile, open ended design is full of hooks so you can intercept and plug-in special handlers to customize or add features to most routines.

VCScreen, our screen painter / code generator speeds your development even more! Simply draw your input forms using our interactive design editor and generate perfect C source code ready to compile and link with the Vitamin C library.

- ☑ Professional C function library
- ☑ 30 day money back guarantee
- ☑ Multiple bullet proof windows
- ☑ Easy full screen data entry
- ☑ Unlimited data validation
- ☑ Context sensitive help manager
- ☑ Menus like Lotus and Mac
- ☑ Programmable keyboard handler
- ☑ Text editor routines
- ☑ No royalties or runtime fees
- ☑ Library source included FREE
- ☑ Free technical support
- ☑ Free BBS at (214)418-0059
- ☑ Supports all major compilers including Microsoft 5.0
- ☑ VCScreen code generator too!
- ☑ UNIX version avaialable, call for details

Vitamin C $225
Includes all source code FREE! For IBM PC, XT, AT, PS/2 and true compatibles. Specify compiler when ordering.

VCScreen $99.95
For IBM PC, XT, AT, PS/2 and true compatibles. Requires Vitamin C.

We ship UPS. Please include $3 for ground, $6 for 2-day air, $20 for overnight, or $30 if outside the U.S. Texas residents add 7 1/4 % sales tax. All funds MUST be in U.S. dollars drawn on a U.S. bank. Visa & MasterCard accepted.

ORDER NOW!

(214)416-6447

creative PROGRAMMING

Box 112097 • Carrollton, Texas 75011

Windows • Data Entry • Menus • Help • Text Editing

Plus... All Source Code FREE!

Listing One

(Listing continued, text begins on page 116.)

```
        /* Print the line if required. The first test is the
         * normal situation (today or tomorrow). The second
         * test takes care of Saturday, which must include
         * the following Monday in its definition of "tomorrow."
         * The third test takes care of December 31, which must
         * recognize January 1 as "tomorrow." Note that I'm
         * intentionally not testing for the years being different
         * because it seems reasonable that, if no year is
         * specified and today is December 31, that the next year
         * is implied by January 1, with no date.
         */

        if( t->tm_year == year &&
                (day == t->tm_yday || day == t->tm_yday +1) )
            fputs( buf, stdout );

        if( t->tm_wday == 6  &&  day == t->tm_yday + 2 )
            fputs( buf, stdout );

        if( t->tm_mday == 31 && t->tm_mon == 11 && day == 0 )
            fputs( buf, stdout );
    }
}
```

End Listing One

Listing Two

```
Listing 2 -- dateh.c, Printed 7/26/1987
```

```
#include <stdio.h>
#include <dos.h>
#include <time.h>
#include <sys/types.h>
#include <sys/stat.h>

/*-------------------------------------------------------------
 *
 *      Creates or updates file called date.h that looks like:
 *
 *              #define _DAY_      "6-17-87"
 *              #define _TIME_24_  "12:5:23"
 *              #define _TIME_     "12:5 PM"
 *              #define _DATE_     "6-17-87 12:5 PM"
 *
 * and holds the current time and date. The file is  created
 * if it doesn't exist. If it does exist, it is modified, but
 * only if it was not modified or created earlier on the same
 * day. All of this behaviour can be modified by command-line
 * switches [see usage(), below].
 *
 * This program uses the Unix time functions. For them to work,
 * you need to set the various codes in the TZ environment
 * variable. For example:
 *              setenv TZ=PST8PDT               (Unix or Sh)
 *              set    TZ=PST8PDT               (COMMAND.COM)
 *
 * Says the Pacific Standard Time is 8 hours off of Greenwich
 * and the PST is used as the abbreviation for it. PDT is used
 * as an abbreviation for daylight savings time.
 *
 * Exit status is 0 if the file is untouched, 1 if it's created
 * or modified, 2 on an error of some sort.
 */

main( argc, argv )
char    **argv;
{
    FILE        *fd;
    struct stat stats;
    struct tm   *t;
    int         day ;
    long        thetime;
    int         pm;
    char        *p;

    int         verbose  = 0;
    int         force    = 0;
    int         create   = 0;

    if( argc != 1 )
    {
            p = argv[1];

            if( *p != '-' )
                usage();
            else
                for( ++p ; *p ; p++ )
                    switch( *p )
                    {
                    case 'v': verbose = 1;      break;
                    case 'f': force   = 1;      break;
                    case 'c': create  = 1;      break;
                    default : usage();
                    }
    }

    thetime = time(NULL) ;

    if( stat("date.h", &stats) != 0 )
    {
        if( !create )               /* File doesn't exist */
            exit( 0 );

        t = localtime( &thetime );
        printf("Creating DATE.H: %s\n", asctime(t) );
    }
    else
    {
        t = localtime( &(stats.st_mtime) );

        if( verbose )
            printf( "DATE.H last modified: %s", asctime(t) );

        day = t->tm_yday;

        t = localtime( &thetime );

        if( verbose )
            printf( "Today is: %s", asctime(t) );

        if( t->tm_yday == day && !force )
            exit( 0 );
        else
            printf("Modifying DATE.H\n");
    }

    if( !(fd = fopen( "date.h", "w" )) )
    {
            perror( "date.h" );
            exit( 2 );
    }
    else
    {
        fprintf(fd, "#define _DAY_      \"%d-%d-%d\"\n",
                                        t->tm_mon,
                                        t->tm_mday,
                                        t->tm_year );

        fprintf(fd, "#define _TIME_24_  \"%d:%d:%d\"\n",
                                        t->tm_hour,
                                        t->tm_min,
                                        t->tm_sec );

        pm = t->tm_hour >= 12;

        if( t->tm_hour > 12 )
                t->tm_hour -= 12;

        else if( t->tm_hour == 0 )
                t->tm_hour  = 12;

        fprintf(fd, "#define _TIME_     \"%d:%d %s\"\n",
                                t->tm_hour,
                                t->tm_min,
                                pm ? "PM" : "AM" );

        fprintf(fd, "#define _DATE_     \"%d-%d-%d %d:%d %s\"\n",
                                t->tm_mon,
                                t->tm_mday,
                                t->tm_year,
                                t->tm_hour,
                                t->tm_min,
                                pm ? "PM" : "AM" );
        fclose(fd);
    }

    exit( 1 );
}

/*-------------------------------------------------------------*/

#define E(x) fprintf(stderr,x)

usage()
{
    E("Usage dateh [-fvc]\n");
    E("If date.h doesn't exist, create it, otherwise if\n");
    E("the date stamp isn't today, update it\n");
    E("\n");
    E("-f  forces an update, even if the date stamp is ok\n");
    E("-v  verbose operation\n");
    E("-c  create file if it doesn't exist\n");
    E("\n");
    exit( 2 );
}
```

End Listings

Breakthrough in interface management. Generate C code from Dan Bricklin's Demo screens. Date fields. Full color support. Money fields. Fully programmable field behavior. Scrolling text within fields. Calculator style numeric input. User definable entry validation. Field marking. Orthogonal field movement. Specify fields by number or location. Source code included. Screen sizes limited only by memory. Interfaces with db__VISTA and other libraries. Text style numeric input. Input masking. List fields. Create spreadsheets. Includes Look & Feel screen designer. Integer fields. String formatting commands. Date and time validation functions. Generate C code with Look & Feel screen designer. Supports automatic vertical and horizontal scrolling. Clean screen machine. Number of fields per screen limited only by memory. Fast screen development. String fields. Easy to modify. Fast screen painting. Bind as much data as de sired to fields. Numeric data entry with commas. Ask a bout our linear pro gramming library. Hexadecimal fields. Long fields. Yes or No fields. Float fields. Quick C. Read only fields. Speaker functions. Lattice. Create reports. Codename Slug. Numeric validation routines. Validate data at the keystroke level. Customize screens and menus at run time. 30 day money back guarantee. Gen eric data pointer. Rich assortment of editing commands. Easy to learn. Pop-up windows. Assign validation data to fields. Corporate C credentials. Pull down menus. Sup ports EGA 43 line mode. All functions are kept in separate modules. Full C style function reference. Pop-up prompt and message functions. Numeric range checking. No royalties. Descrip- tive function names. Date and time conversion routines. Capture screens from existing programs. Nest screens as deep as desired. Easy to main tain. Run time error checking. Date and time conver sion functions. Screen definition language based on C's printf. Time fields. Ful- ly definable borders. The current field can be automati- cally highlighted. Create reports. Exploding borders. Convert old programs to C. Borders with titles. Color map enables use of logical colors. Toll-free telephone support line. 24 hour bulletin board. Automatically detects type of monitor being used. ANSI driver included. Screen and field definitions. Uses device drivers for portability. View text in pop-up windows. Read only fields. Rich assortment of editing commands. Pass- word entry fields. Includes ROM BIOS driver. Fields can support any data type. Scrolling/ paging functions included. Specify writeable and non-writeable positions within fields. Customizable borders. Se lect different cur sor types. Supports CGA, EGA, and monochrome. Aztec. In cludes functions for writing to the display. In cludes an ANSI device dri ver. A vari ety of keyboard functions. Lined borders. Multi-level menuing systems. Borders with scroll lights. Vid eo RAM dri ver included. New device drivers can be cre ated. Color map enables log ical use of colors. Borders with prompt lines. Fully inte grated help system. Create as many screens as needed. Create data entry screens. Easy to follow man ual. Professional customer sup port. Includes higher level functions. Device drivers swappable at run-time. Context sensitive help system. Cross referenced help screens. Protected fields. Object-oriented design. Read in screen defini tions from disk files. Digitally mastered. Assign prompt strings to fields. UNIX. No run- time license. Numeric range checking. Unified field theory. Full printf % substitution within screen definitions. Supports all memory models. C Bricklin run. Turbo C. 24 hour bulletin board. Higher level functions included. Object-oriented design. All library functions are kept in separate modules. Nest screens as deep as desired. Design screens with Look & Feel screen designer. New device drivers can be created. Create as many screens as needed. No run-time license. Hexadecimal fields. Prefer- red by professionals and consultants. Microsoft. Cross referenced help system. by space aliens. Generate C code with Look & Feel screen designer. Context sensitive help system. Scroll lights. Read in screen definitions from disk files. Automatic vertical and horizontal scrolling. Batteries not included. Double and single line borders. Cross-referenced help system. Save and restore regions of the display. Nested menus. Quick C. Create screens from ASCII files. Easy to learn and use. Horizontal and vertical scrolling. Used by consultants and corporations worldwide. Easy to maintain. Professional documentation. Screen designer creates readable C code. Portable. Easily modifiable functions. No royalties. Source code included. Turn Dan Bricklin slides into C. Professional support. Interface examples for data base management. Validation at keystroke level. Vast integrated and indexed context-sensitive help system. Save and restore regions of the display. Now supporting Quick C, Turbo C, Aztec, Lattice, Microsoft, UNIX and others. And that's not all. Call for demo.

C-scape 2.0

NEW!

with

Look & Feel™

The state-of-the-art interface management system preferred by professional C programmers and consultants worldwide.

Look & Feel

- WYSIWYG screen design tool
- Generates readable C code
- Create menus and data entry screens
- Define fields of any type
- Variables, prompts, and validation
- Line draw and erase
- Block, move, cut, paste, copy
- Horizontal and vertical scrolling
- Edit Dan Bricklin Demo slides
- Full color support
- Fast, easy, and fun to use
- Includes help
- Full-feature demo available

C-scape 2.0

- Windows, windows, windows
- Menus, menus, menus
- Vast help system
- Create any type of field
- Data entry and validation
- Smart borders
- Extensive function library
- Swappable device drivers
- Easy to learn and use
- Easy to maintain and modify
- Unsurpassed flexibility
- Professional manual
- No royalties; no run-time license
- Source code included
- Demo package available

Oakland Group, Inc.

675 Massachusetts Avenue
Cambridge, MA 02139-3309

800-233-3733 CALL
617-491-7311 NOW

MasterCard VISA

PC/MS-DOS $279, plus shipping (includes C-scape, Look & Feel, source, manual and support). UNIX/others call. 30-day review.

CIRCLE 227 ON READER SERVICE CARD

Listing One *(Text begins on page 124.)*

```
Listing 1.  Pascal function for the simple heuristic search in an
            unordered array.

FUNCTION Search1(VAR Data   : AnyArrayType; { input   }
                 VAR Index  : IntegerArray; { in/out  }
                     NData  : INTEGER;      { input   }
                     Item   : ScalarType    { input   }) : INTEGER;

{ function returns the index of the matching
  array, or zero if no match is found          }

VAR I, Tempo : INTEGER;

BEGIN
    I := 1;
    { scan array }
    WHILE (Item <> Data[ Index[I] ]) AND (I <= NData)   DO
            I := I + 1;

    IF I <= NData
    THEN BEGIN { match found }
        Search1 := Index[I]; { returned result }
        { Swap indices ? }
        IF I > 1 THEN BEGIN
            Tempo := Index[I];
            Index[I] := Index[I-1];
            Index[I-1] := Index[I];
        END { IF }
    END
    ELSE Search1 := 0; { not found }

END; { Search1 }
```

End Listing One

Listing Two

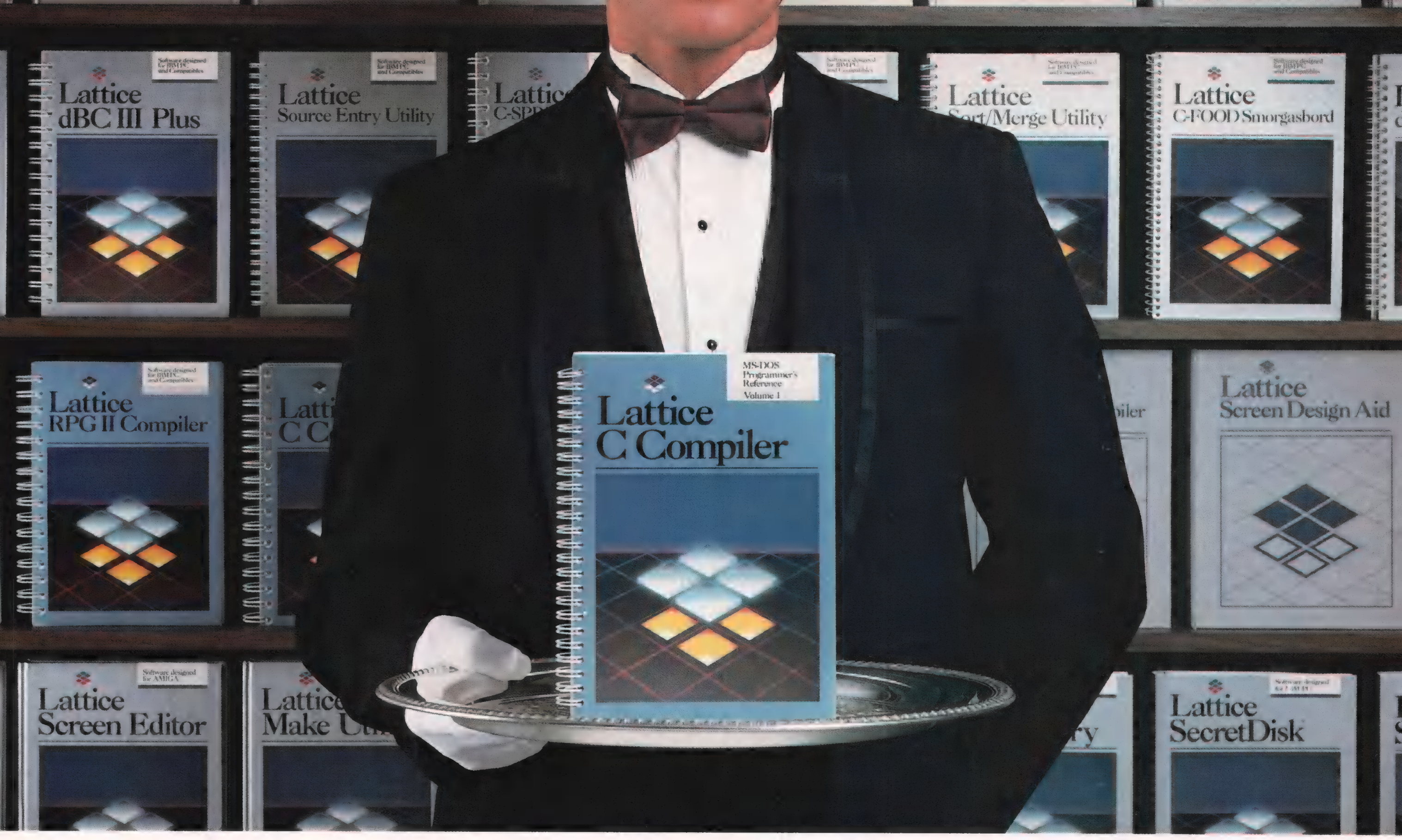
```
Listing 2.  Pascal function for the heuristic search method for
            unordered arrays.

FUNCTION Search2(VAR Data   : AnyArrayType; { input   }
                 VAR Index  : IntegerArray; { in/out  }
                 VAR Loctn  : SmallIntArray;{ in/out  }
                     NData  : INTEGER;      { input   }
                     Item   : ScalarType    { input   }) : INTEGER;

{ function returns the index of the matching
  array, or zero if no match is found          }

CONST Factor = 0.7; { time-series factor }
      { limits used to select search schemes }
      UPPER_LIMIT = 0.65; { = 0.5 + 0.15 }
      LOWER_LIMIT = 0.35; { = 0.5 - 0.15 }
      { number of Loctn elements used in predicting the next
        matching location }
      N = 4;

VAR Continue : BOOLEAN;
    I, J, Tempo, This_Location, Median,
    Skip, Result : INTEGER;
    Next_Location, Power : REAL;


PROCEDURE Swap_Indices(K : INTEGER);
{ Procedure used to swap indices }
BEGIN
    { Swap indices ? }
    IF K > 1 THEN BEGIN
        Tempo := Index[K];
        Index[K] := Index[K-1];
        Index[K-1] := Index[K];
    END { IF }
END;


BEGIN
    { Estimate the next location }
    Next_Location := 0.0; { initialize next location }
    Power := 1.0;
    FOR I := N DOWNTO 1 DO BEGIN
        Next_Location := Next_Location + Power * Loctn[I];
        Power := Power * Factor;
    END;
    { calculate predicted next search location as a fraction }
    Next_Location := (1.0 - Factor) * Next_Location   / NData;


    Result := 0; { default value for no match }

    IF Next_Location > UPPER_LIMIT THEN BEGIN
        { Search last-to-first }
        I := NData;
        WHILE (Item <> Data[ Index[I] ]) AND (I > 0)   DO
            I := I - 1;

        IF I > 0 THEN BEGIN
            Result := Index[I];
            Swap_Indices(I); { swap indices }
        END { IF }
    END
    ELSE IF Next_Location < LOWER_LIMIT THEN BEGIN
        { Search first-to-last }
        I := 1;
        WHILE (Item <> Data[ Index[I] ]) AND (I <= NData)   DO
            I := I + 1;
        IF I <= NData THEN BEGIN
            Result := Index[I];
            Swap_Indices(I); { swap indices }
        END { IF }
    END
    ELSE
        { Perform bidirectional search starting at the middle }
        Median := NData div 2;
        IF Item <> Data[ Index[Median] ] THEN BEGIN
            Skip := 1;
            Continue := TRUE;
            REPEAT
                I := Median + Skip;
                IF I <= NData THEN
                    IF Item = Data[ Index[I] ] THEN BEGIN
                        Result := Index[I];
                        Continue := FALSE;
                        Swap_Indices(I); { swap indices }
                    END; { IF }

                J := Median - Skip;
                IF J > 0 THEN
                    IF Item = Data[ Index[J] ] THEN BEGIN
                        Result := Index[J];
                        Continue := FALSE;
                        Swap_Indices(J); { swap indices }
                    END; { IF }

                IF (I > NData) AND (J < 1) THEN { out of bounds }
                    Continue := FALSE;

                Skip := Skip + 1;
                UNTIL (NOT Continue);

            END
            ELSE BEGIN
                Result := Index[Median];
                Swap_Indices(Media)
            END; { IF }
        END; { IF }

        IF Result > 0 THEN BEGIN
            { Update location array }
            FOR I := 1 TO N-1 DO
                Loctn[I] := Loctn[I+1];
            Loctn[N] := Result
        END; { IF }

        Search2 := Result; { return result }


END; { Search 2 }
```

End Listing Two

Listing Three

```
Listing 3.  Pascal code for function implementing the first
            heuristic search method for fixed ordered arrays.

PROCEDURE Initialize(VAR Data    : AnyArrayType; { input   }
                     VAR Table   : RecordArray;  { in/out  }
                     TableSize   : INTEGER;      { input   }
                         NData   : INTEGER       { input   });

{ initialize index table by inserting data at equal intervals }
{

    TYPE TableRecord = RECORD
            Key : ScalarType;
            Index : INTEGER;
         END;

         RecordArray = ARRAY[1..MAX_TABLE_SIZE] OF TableRecord;
}

VAR I, J, Delta : INTEGER;

BEGIN
    Delta := NData div TableSize;
    J := 1;
    FOR I := 1 TO TableSize DO BEGIN
        Table[I].Key := Data[J];
        Table[I].Index := J;
        J := J + Delta;
    END;

END; { initialize }



FUNCTION Search3(VAR Data   : AnyArrayType; { input   }
                 VAR Table  : RecordArray;  { in/out  }
                 TableSize,                 { input   }
                     NData  : INTEGER;      { input   }
                     Item   : ScalarType    { input   }) : INTEGER;

VAR Found, NoMatch : BOOLEAN;
    First, Last, I, K, Result : INTEGER;
```

(continued on page 115)


Lattice dBC III Plus
Lattice Source Entry Utility
Lattice Sort/Merge Utility
Lattice C-FOOD Smorgasbord
Lattice RPG II Compiler
MS-DOS Programmer's Reference Volume 1
Lattice C Compiler
Lattice Screen Design Aid
Lattice Screen Editor
Lattice SecretDisk

Our software comes with something no one else can offer.

Lattice Service.

When you join the Lattice family of customers, you'll discover that your software purchase is backed by more than just an excellent warranty. It's backed by unparalleled technical support. By a total commitment to your success and satisfaction. And by Lattice's dedication to excellence in products and services.

Unlike other software manufacturers who charge you for services after you've purchased their product, Lattice offers a unique package of support programs at a price we can all live with—FREE.

Lattice Bulletin Board Service
LBBS is our 24-hour a day bulletin board system that allows you to obtain notification of new releases, general information on Lattice products, and programs for the serious user. And if you've ever experienced the frustration of having to wait a year or more for a new release (that has corrected a bug), you'll really appreciate LBBS. Because with this service, you can actually download the latest program fixes to instantly eliminate any bugs discovered after release.

Technical Support Hotline
Responsible, dependable and capable Support Representatives are only a phone call away. You will talk to a highly skilled expert who is trained to answer any questions you have relating to specific Lattice products. Remember, your complete satisfaction is our goal.

McGraw-Hill BIX™ Network
The Byte Information Exchange (BIX) Network is a dial-in conference system that connects you with a Special Interest Group of Lattice users. The nominal one-time registration fee allows you to BIX-mail your questions—via your modem—directly to Lattice. Or you can post your questions in the conference mode for Lattice or other users to answer. Once again, you have 24-hour access.

You Also Receive:
▪ Timely updates and exciting enhancements ▪ 30-day, money-back guarantee ▪ Lattice Works Newsletter ▪ Technical Bulletins ▪ Access to Lattice User Groups

Lattice has developed more than 50 different Microcomputer software tools that are used by programmers worldwide. We were there for every MS-DOS release. We're there now for OS/2. And we'll be there for the next generation of technical changes. But most of all, Lattice is there for you.

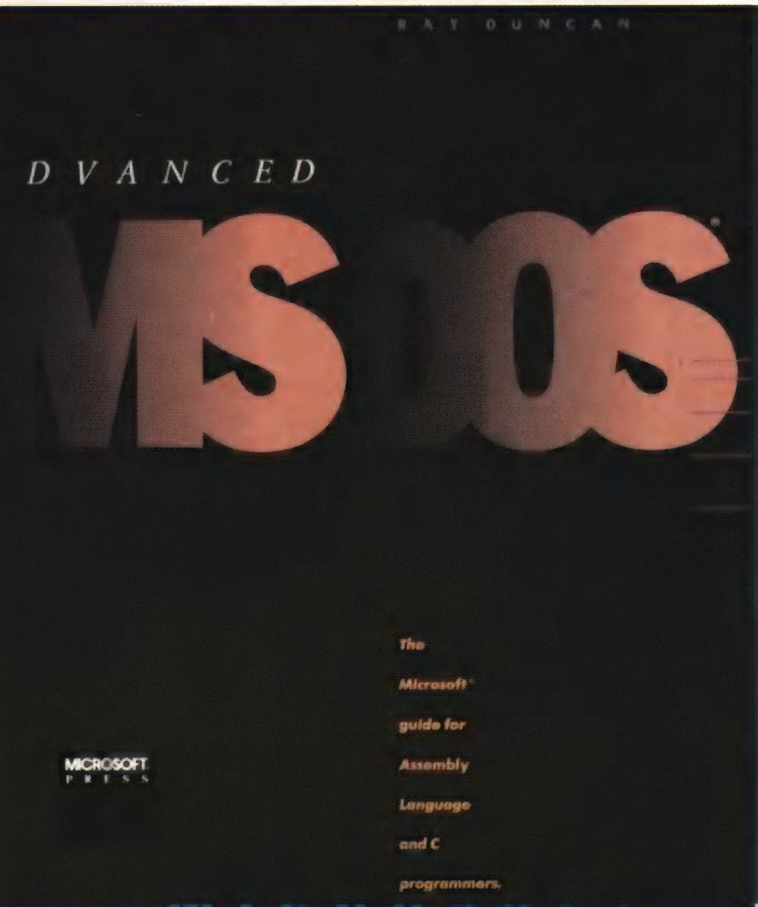
Lattice
Subsidiary of SAS Institute Inc.

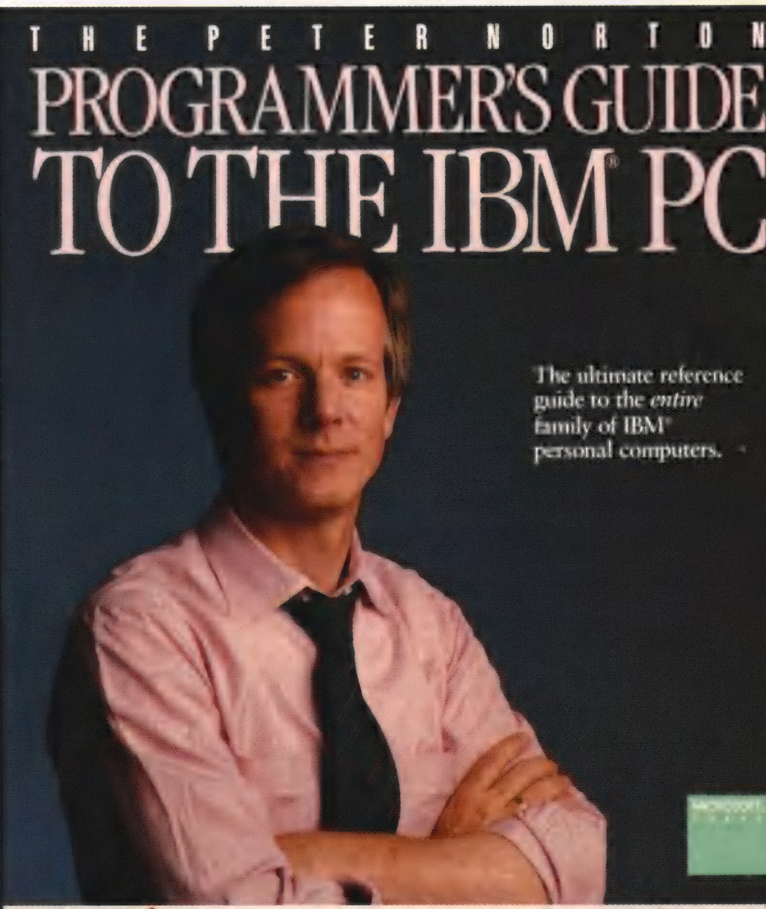
Lattice, Incorporated
2500 S. Highland Avenue
Lombard, IL 60148
Phone: 800/533-3577
In Illinois: 312/916-1600

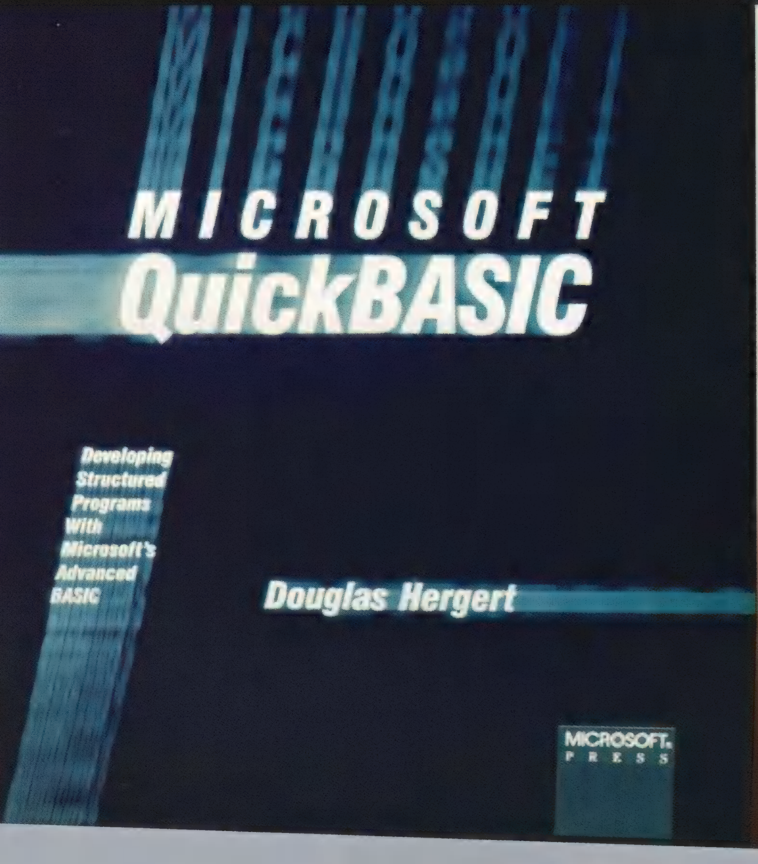
Available through dealers and distributors worldwide.

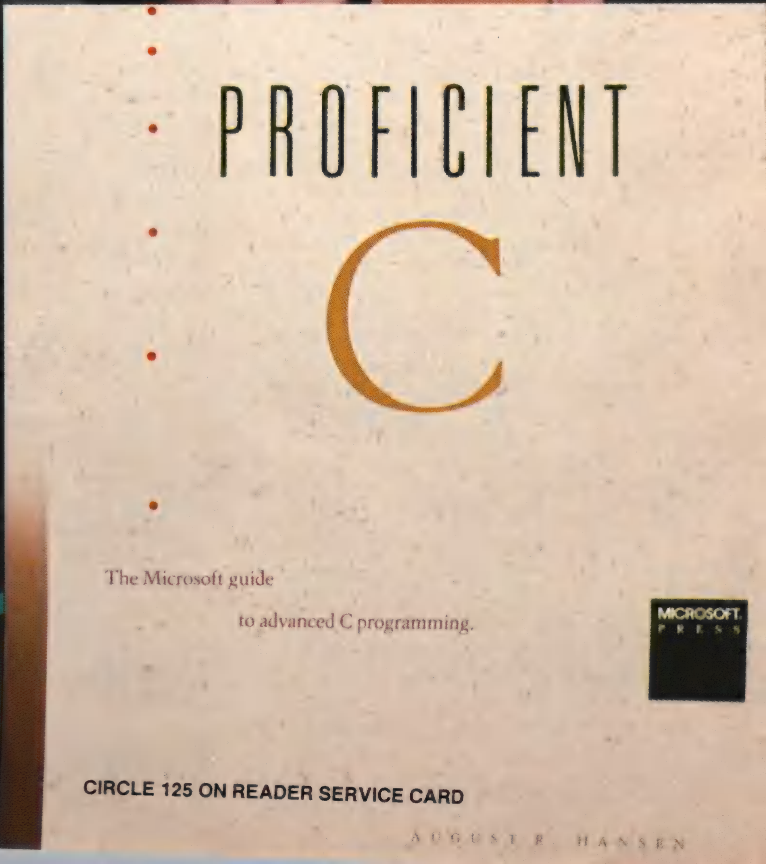
CIRCLE 101 ON READER SERVICE CARD

Think of us as the Book of the Mind Club.


RAY DUNCAN
DVANCED
MS DOS
The Microsoft guide for Assembly Language and C programmers.
MICROSOFT PRESS


THE PETER NORTON
PROGRAMMER'S GUIDE
TO THE IBM PC
The ultimate reference guide to the entire family of IBM personal computers.
MICROSOFT PRESS


MICROSOFT
QuickBASIC
Developing Structured Programs With Microsoft's Advanced BASIC
Douglas Hergert
MICROSOFT PRESS



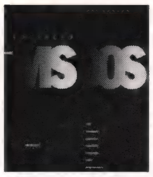
CIRCLE 125 ON READER SERVICE CARD

Want to turn programming time into prime time? Want to put some topspin on your techniques? Want to develop invaluable new resources?

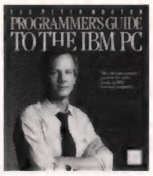
Time to hit the books. From Microsoft® Press. The best and brightest books in the business.

Our parent company is Microsoft, the folks who taught the PC how to think. Our authors read like a Who's Who of What's What.

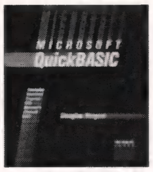
Here are four ways to boost your computer's I.Q.:



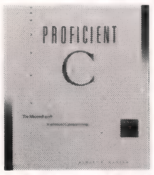
Advanced MS-DOS® by Ray Duncan. Also known as an information bonanza for assembly language and C programmers. Disk files, records, directories, volume labels, internals, memory management, EXEC functions, installable device drivers. More. Ray Duncan has it down. Now you can, too. $22.95. 468 pages. Softcover.



The Peter Norton Programmer's Guide to the IBM® PC by Peter Norton. Want to develop intermediate and advanced programs you can port from one branch of the PC tree to another? Want to understand the hardware? Software? The differences between PC, XT, AT and Jr.? Get the latest tech talk? Relax. The leading authority in the field leads you out of the bog. $19.95. 448 pages. Softcover.



Microsoft QuickBASIC by Douglas Hergert. Here's the perfect way to get up to speed with QuickBASIC. Plus five, smart, sample programs that'll tweek your QuickBASIC skills: MORTGAGE, for data types; QUICKCHART, for graphics; SURVEY, for data-file techniques; EMPLOYEE, for random-access files; TWENTY-ONE, for IF...THEN...ELSE games. $18.95. 384 pages. Softcover.



Proficient C by Augie Hansen. Cross DOS and C and what do you get? Powerful programs that run at warp speed. Use the ANSI.SYS device driver and the MAKE and LIB utilities to learn valuable, reuseable methods of structured program development. From the man whose proficiency at Bell Labs, General Dynamics and Raytheon was the springboard to this expert guide for intermediates—and experts. $22.95. 512 pages. Softcover.

Don't fumble for answers. Turn to Microsoft Press. Remember: What you get out of your PC depends on what you read into it.

Available wherever books and software are sold. Credit card orders call 1-800-638-3030. In Maryland call collect, 824-7300.

Microsoft PRESS

Microsoft and MS-DOS are registered trademarks of Microsoft Corporation. IBM is a registered trademark of International Business Machines Corporation.

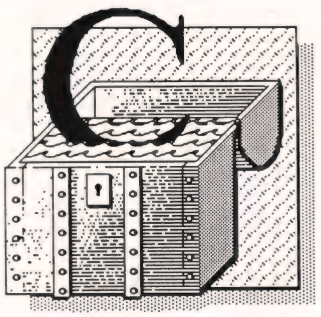
CIRCLE 125 ON READER SERVICE CARD

Listing Three *(Listing continued, text begins on page 124.)*

```
BEGIN
    Result := 0; { initialize result with default value }

    { Search for Item in index table }
    I := 1;
    WHILE (Item > Table[I].Key) AND (I <= TableSize) DO
        I := I + 1;

    Found := FALSE;

    IF I <= TableSize
        THEN Found := (Item = Table[I].Key)
        ELSE I := I - 1;

    IF Found
    THEN
        Result := Table[I].Index
    ELSE
        { Get range for search limits }
        First := Table[I-1].Index;
        Last := Table[I].Index;
        K := I-1;
        NoMatch := TRUE;
        I := First;
        WHILE (I <= Last) AND NoMatch DO
            IF Item <> Data[I].Key THEN I := I + 1
                                   ELSE NoMatch := FALSE;


        IF NOT NoMatch THEN BEGIN
            Result := Table[I].Index; { store result }

            IF K > 1 THEN BEGIN { update table entry }
                Table[K].Key := Item;
                Table[K].Index := Result
            END; { IF }

        END; { IF }

    END; { IF }

    Search3 := Result { return result }

END; { Search3 }
```

End Listings

Using the Unix/ANSI Time Functions



by Allen Holub

This month I'm going to look at the Unix (and now ANSI) time functions and demonstrate their use with two short but useful programs—calendar and dateh.

Calendar is an implementation of the Unix program of the same name. It searches a file called calendar and prints every line that starts with today's or tomorrow's date. If you put it into your AUTOEXEC.BAT file and put the calendar in your root directory, the program prints reminders of what you have to do on any given day when you start up your system.

The second program, dateh, creates (or modifies) a file called date.h to hold C *#define*s for a few strings that represent today's date and the time when dateh was run. You can use these *#define*s in a program's sign-on message to keep track of when the program was compiled. A sample dateh output file is shown in Example 1, below. The file is modified only if it already exists and if it hasn't been modified earlier today, though you can change this behavior with command-line switches. The default behavior facilitates using dateh along with a make utility. You can call it every time you compile, but you'll only have to recompile the file that *#include*s date.h once a day. Dateh returns an exit status of 1 if the file is modified (0 if it's not, 2 on a command-line syntax error). So, you can use the COMMAND.COM *ERRORLEVEL* mechanism (or my shell's[1] *$status* variable) in a batch file or shell script to decide whether or not to recompile.

The Time Functions

The Unix time functions provide a portable way to access the time and date from within a program. You can use them to get both time and date during execution of a program and the time-and-date stamp associated with a file. Because they have been incorporated into ANSI, they provide a degree of system independence not available if you do direct DOS calls. All these functions require an *#include <time.h>* at the top of your program.

There are two classes of time functions. The lowest-level function, called *time( )*, provides the time and date as the number of seconds elapsed since midnight, January 1, 1970 GMT. The returned value is of type *long*. Because there are roughly 31,557,600 seconds in a year (365.25 ×24×60×60), and because the largest (signed) 32-bit number that a *longint* can hold is 2,147,483,647, the time will roll over on January 18, 2038 at 2:56:02 a.m. This means, of course, that all Unix programs that use *time( )* will fail on in the morning of January 18, 2038—you can't have everything.

The ANSI standard differs from Unix in that the return value from *time( )* is an object of type *time_t* rather than *long* and *time_t* is left undefined. This means that a compiler manufacturer could *typedef time_t* as a *double*, for example, in order to move the rollover point well into the next century. I'll use the ANSI convention for the rest of this article. The ANSI calling syntax for *time( )* is:

```
time_t time( timeptr );
time_t *timeptr;
```

and the Unix syntax is:

```
long time( timeptr );
long *timeptr;
```

The *time( )* function always returns the current time. In addition, if *timeptr* is not *NULL*, it will also load the time into the object pointed at by *timeptr*. The function returns *((time_t)(–1))* if the calendar time isn't available.

A second low-level function that's useful in conjunction with the *time( )* function is *stat( )*. The calling syntax is:

```
#include <sys/types.h>
#include <sys/stat.h>

int stat( path, buf )
char *path;
struct stat *buf;
```

The first argument is the path name of a file; the second is a pointer to a structure declared in stat.h. *Stat( )* fills the structure with information about the file, such as the permission mask, file size, and so forth. There are three fields that are time related: *st_atime, st_mtime,* and *st_ctime.* These hold the file's date stamp, and they use the same elapsed-time-since-1970 mechanism that is used by *time( )*. In Unix systems, *st_atime* holds the time of last access, *st_mtime* is the time

```
#define _DAY_      "6-17-87"
#define _TIME_24_  "13:11:41"
#define _TIME_     "1:11 PM"
#define _DATE_     "6-17-87 1:11 PM"
```

Example 1 *Date.h file created by dateh*

Announcing - the database development system that you designed.

™

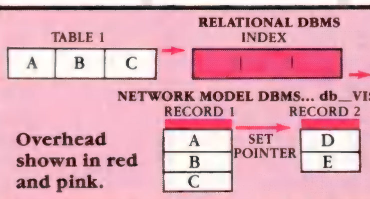
db_VISTA III

C PROGRAMMERS-
We asked what you wanted in a database development system and we built it!

db_VISTA III™ is the database development system for programmers who want powerful, high performance DBMS capabilities ... and in any environment. Based on the network database model and the B-tree indexing method, db_VISTA III gives you the most powerful and efficient system for data organization and access. From simple file management to complex database structures with millions of records. db_VISTA III runs on most computers and operating systems like MS-DOS, UNIX, VAX/VMS and OS/2. It's written in C and the complete source code is available, so your application performance and portability are guaranteed! With db_VISTA III you can build applications for single-user microcomputers to multi-user LANs, up to minis and even mainframes.

RAIMA'S COMMITMENT TO YOU: No Royalties, Source Code Availability, 60 days FREE Technical Support and our 30-day Money-Back Guarantee. Extended services available include: Application Development, Product Development, Professional Consulting, Training Classes and Extended Application Development Support.

HOW TO ORDER: Purchase only those components you need. Start out with Single-user for MS-DOS then add components, upgrade ... or purchase Multi-user with Source for the entire db_VISTA III System. It's easy... call toll-free today!



Overhead shown in red and pink.

db_VISTA out-performs relational DBMS's with direct access to data and less data redundancy. Your applications run faster and more efficiently!

The db_VISTA III™ Database Development System

1 db_VISTA™: The High Performance DBMS

The major features include:

- Multi-user support for LANs and multi-user computers.
- Multiple database access.
- File and record locking.
- Automatic database recovery.
- Transaction processing and logging.
- Timestamping.
- Database consistency check utility.
- Fast access methods based on the network database model and B-tree indexing. Uses both direct "set" relations and B-tree indexing independently for design flexibility and performance.
- An easy-to-use interactive database access utility.
- File transfer utilities for importing/exporting ASCII text and dBASE II/III files.
- A Database Definition Language patterned after C.
- Virtual memory disk caching for fast database access.
- A runtime library of over 100 functions.
- **Operating systems:** MS-DOS, UNIX V, XENIX, VMS, OS/2.
- **C Compilers:** Lattice, Microsoft, IBM, Aztec, Computer Innovations, Turbo C, XENIX, and UNIX.
- **LAN systems:** LifeNet, NetWare, PC Network, 3Com, SCO XENIX-NET, other NETBIOS compatible MS-DOS networks.

2 db_QUERY:™ The SQL-based Query.

- Provides relational view of db_VISTA applications.
- Structured Query Language
- C linkable.
- Predefine query procedures or run ad-hoc queries "on the fly".

3 db_REVISE™: The Database Restructure Program.

- Redesign your database easily.
- Converts all existing data to revised design.

All components feature royalty-free run-time distribution, source code availability and our commitment to customer service. That's why corporations like ARCO, AT&T, Hewlett-Packard, IBM, Northwestern Mutual Life, UNISYS and others use our products.

db_VISTA III™ Database Development System

db_VISTA III™	$595 - 3960
db_QUERY™	$595 - 3960
db_REVISE™	$595 - 3960
db_VISTA™ File Manager	Starts at $195

We'll answer your questions, help determine your needs and get you started.

CALL TODAY!
1-800-db-RAIMA
(that's 1-800-327-2462)

MasterCard VISA

RAIMA™ CORPORATION

3055 112th Avenue N.E., Bellevue, WA 98004 **(206)828-4636**
Telex: 6503018237MCIUW FAX: (206)828-3131

DD 117

C CHEST

(continued from page 116)

when the file was last modified, and *st_ctime* is the create time. In MS-DOS compilers such as Microsoft's, all three fields are set to the last-modified time because that's the only information available from DOS.

A related function, *fstat( )*, works rather like *stat( )* does, but it takes a *FILE* pointer as its first argument rather than a path name.

The *difftime( )* function returns the difference between two times, as returned by *time()* or *stat( )*. Its calling syntax is:

```
double difftime (t1, t2)
time_t t1, t2;
```

Note that the returned value is of type *double*, not *time_t*. *Difftime( )* is necessary because you do not know the actual type of a *time_t* (it may be a structure, for example). Consequently, you can't just subtract two times to get the difference if you want your code to be portable.

Note that the minimum resolution available from *time( )* is seconds. If you need fractional time, you need to use the *ftime( )* function. Its calling syntax is:

```
#include <sys/types.h>
#include <sys/timeb.h>

void ftime( timep )
struct timeb *timep;
```

The *timeb* structure is shown in Example 2, below. The two fields of interest are the *time* field, which holds the same value as would be returned from *time( )*, and the *millitm* field, which holds an additional number of milliseconds (thousandths of a second). Note that the resolution here is not better than your system clock. For example, the IBM PC clock ticks 18.2 times a second, so the maximum resolution on a PC is 54 milliseconds.

The second level of time functions all represent the time as a structure, also declared in time.h (and shown in Example 3, below).

Two functions—*gmtime( )* and *localtime( )*—convert the *time_t* times, as returned by *time( )* or *stat( )*, into *tm* structures. They both have the same calling syntax:

```
struct tm *gmtime ( time )
struct tm *localtime( time )

time_t *time_ptr;
```

Gmtime( ) converts to Greenwich Mean Time, and *localtime( )* goes to the current local time. Note that you're passing in a pointer to a variable that holds the time, not the time itself.

The local time is determined in mysterious ways, depending on your system. Most Unix systems have it hard-coded into the *localtime( )* function, which automatically figures out the peculiarities of daylight savings time and so forth. Many compilers, however, use an environment variable to determine the local time. For example, Microsoft's uses an environment variable called *TZ* for this purpose. It's set to a string something like *PST8PDT*, which says that Pacific Standard Time (PST) is 8 hours off Greenwich and that Pacific Daylight Time (PDT) is 1 hour off that. Microsoft provides a *tzset( )* function that can be used to set this environment variable from within a program. In Unix, the name of the current time zone is available through the *timezone( )* function. Consult the ctime(3) section of the manual for details.

The *mktime( )* function goes in the other direction. It takes as input a *tm* structure and returns the equivalent time in a *time_t*, as would have been returned by *time( )*. The calling syntax is:

```
time_t mktime(time_ptr)
struct tm *time_ptr;
```

Mktime( ) can also be used to flush out the *tm_mday* and *tm_yday* fields of the structure. That is, if you want to find out the day of the week for a particular day, you can fill up a *tm* structure yourself, leaving the *tm_wday* and *tm_yday* fields empty, and then *mktime( )* will modify those fields as required.

Three functions are provided for printing the time: *strftime( )*, *ctime( )*, and *asctime( )*. *Strftime( )* is an ANSI function, not supported by Unix, that works something like *sprintf( )* does. Because it's not implemented by most compilers, I won't discuss it further. Calling syntaxes for the other two functions are:

```
char *asctime( time )
struct tm *time;

char ctime( tp )
time_t *tp;
```

```
struct timeb
{
  time_t         time;     /* Time, like that returned from time().  */
  unsigned short millitm;  /* fraction of a second, in milliseconds */
  short          timezone; /* The difference moving westward, in    */
                           /* minutes, between Greenwich Mean Time  */
                           /* and the local time.                   */
  short          dstflag;  /* Nonzero if daylight savings time is   */
                           /* in effect.                            */
};
```

***Example 2** The* timeb *structure*

```
struct tm
{
    int tm_hour;        /* hour                          */
    int tm_min;         /* minute                        */
    int tm_sec;         /* second                        */
    int tm_isdst;       /* daylight savings time is active */
    int tm_year;        /* year - 1900                   */
    int tm_mon;         /* month; January == 0           */
    int tm_mday;        /* day of the month              */
    int tm_wday;        /* day of the week; Sunday == 0  */
    int tm_yday;        /* number of elapsed days since  */
                        /* January 1, January 1 == 0    */
}
```

***Example 3** A* tm *structure*

They both return a string that is exactly 26 characters long and has the following form:

```
Wed Jul 22 11:57:45 1987\n\0
```

Asctime() takes a pointer to a *tm* structure as its argument, as is returned from *gmtime()* or *localtime()*. *Ctime()* takes a pointer to a *time_t* argument, as is returned by *time()* or *stat()*. Note, again, that this argument is a pointer to a variable holding the time, not to the time itself.

Calendar.c

Calendar is a DOS version of the Unix memo program. It searches a file called calendar, which must be in the current directory, for lines starting with a date, and if that date is either today or tomorrow, it prints the line. Note that this behaviour differs from that of the Unix program, which allows the date to be anywhere on the line. My implementation ignores white space that precedes the date, but no other characters can be present. "Tomorrow" for weekends is extended to include the following Monday.

My calendar, unlike the Unix version, returns an exit status of 0 when it prints something (nonzero if it doesn't). I invoke it from the LOGIN.BAT file used by my shell with the following:

```
calendar
if( $status == 0 ) then
    bell
    bell
endif
```

Bell is a two-line program that rings the bell on the PC. This way I get both the message and an audible signal when I have something to do. You could do the same thing from within a COMMAND.COM batch file by using the *ERRORLEVEL* mechanism.

The program recognizes most forms of dates, though the date must take the form of month, then day, then year; neither *87-9-17* nor *17 Sept* is recognized. All the following are recognized, however:

```
Sept. 7, 1987
September 7 '87
Sep. 7
```

NEW VERSION

PC/VI™

UNIX's VI Editor Now Available For Your PC!

Are you being as productive as you can be with your computer? An editor should be a tool, not an obstacle to getting the job done. Increase your productivity today by choosing **PC/VI**—a COMPLETE implementation of UNIX* VI version 3.9 (as provided with System V Release 2).

PC/VI is an implementation of the most powerful and most widely used full-screen editor available under the UNIX operating system. The following is only a hint of the power behind **PC/VI:**

- Global search or search and replace using regular expressions
- Full undo capability
- Deletions, changes and cursor positioning on character, word, line, sentence, paragraph, section or global basis
- Editing of files larger than available memory
- Shell escapes to DOS
- Copying and moving text
- Macros and Word abbreviations
- Many controllable options including Auto-indent, Showmatch, and Wrap Margin
- Filter text through external programs AND MORE!

Don't take it from us. Here's what some of our customers say: "Just what I was looking for!", "It's great!", "Just like the real VI!". "The documentation is so good I have already learned things about VI that I never knew before." — *IEEE Software*, September 1986.

PC/VI is available for IBM-PC's and generic MS-DOS† systems for only $149. Included are CTAGS and SPLIT utilities, TERMCAP function library, and an IBM-PC specific version which enhances performance by as much as TEN FOLD!

PC/TOOLS™

What makes UNIX so powerful? Sleek, Fast, and **POWERFUL** utilities! UNIX gives the user not dozens, but hundreds of tools. Now the most powerful and popular of these are available for your PC! Each is a complete implementation of the UNIX program. Open up our toolbox and find:

• ASA	• COMM	• DIFF3	• MV	• SEE	• TR
• BANNER	• CP	• FIND	• OD	• SORT	• TOUCH
• BFS	• CUT	• GREP	• PASTE	• SPLIT	• UNIQ
• CAL	• DATE	• HEAD	• PR	• STRINGS	• WC
• CAT	• DIFF	• LS	• RM	• SUM	
• CHMOD	• DIFFH	• MAKE	• SED	• TAIL	

All of these for a limited introductory price of only $49.00; naturally, extensive documentation is included!

PC/SPELL™

Why settle for a spelling checker which can only compare words against its limited dictionary database when **PC/SPELL** is now available? **PC/SPELL** is a complete implementation of the UNIX spelling checker, renowned for its understanding of the rules of English! **PC/SPELL** determines if a word is correctly spelled by not only checking its database, but also by testing such transformations as pluralization and the addition and deletion of prefixes and suffixes. For only $49.00, **PC/SPELL** is the first and last spelling checker you will ever need!

Buy **PC/VI** and **PC/TOOLS** *now* and get **PC/SPELL** for only $1.00! Site licenses are available. Dealer inquiries invited. MA residents add 5% sales tax. AMEX, MC and Visa accepted without surcharge. Thirty day money back guarantee if not satisfied! Available in 5¼", 3½" and 8" disk formats. For more information call today! *UNIX is a trademark of AT&T. †MS-DOS is a trademark of Microsoft.

CUSTOM SOFTWARE SYSTEMS
P.O. BOX 678 • NATICK, MA 01760
617•653•2555

CSS

UNIX TOOLS FOR YOUR PC

UNIX TOOLS FOR YOUR PC

CIRCLE 268 ON READER SERVICE CARD

THE CURE FOR COMMON CODE

Are you getting the recommended monthly allowance of C, Assembly, Forth, Pascal, BASIC or Modula-2? Subscribe to *Dr. Dobb's Journal of Software Tools* and you won't catch any nasty bugs again!

Each month the Doctor brings you aid for ailing algorithms and the cure for common code. For the latest developments in software design and pages of code that will make you a more productive programmer, take the Dr. Dobb's prescription.

For more than a decade, the programming elite have known *Dr. Dobb's Journal* to be the foremost source of software tools. Subscribe now and get your monthly dose from the Doctor.

SINCE 1976
Dr. Dobb's Journal
INGREDIENTS
Articles, Reviews, C Chest, Software Toolbox, Swaines Flames, Structured Programming, Artificial Intelligence, DDJ Online, The Examining Room, Programming Paradigms
PASCAL
PROLOG
FORTH
C
BASIC
ASSEMBLY
MODULA-2

Dr. Dobb's Journal

of Software Tools

The Rx for Programmers

Subscribe Now &

Save Over 15%

Off the Newsstand Price!

$5 SAVINGS

SUBSCRIBE AND SAVE!
Subscribe to

DR. DOBB'S JOURNAL OF SOFTWARE TOOLS
and save over $5—a 15% savings off the cover price!

☐ **1 year $29.97** ☐ **2 years $56.97**

☐ Please charge my: ☐ Visa ☐ Master Card ☐ American Express

☐ Payment enclosed ☐ Bill me later

Card # ____ Exp. date ____

Signature ____

Name ____

Address ____

City ____ State ____ Zip ____

ONLY $29.97! YOU SAVE OVER $5.00

Savings based on a full one-year cover price of $35.40. Canada & Mexico add $10 for surface mail per year. All other countries add $27 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc. **3463**

$5 SAVINGS

SUBSCRIBE AND SAVE!
Subscribe to

DR. DOBB'S JOURNAL OF SOFTWARE TOOLS
and save over $5—a 15% savings off the cover price!

☐ **1 year $29.97** ☐ **2 years $56.97**

☐ Please charge my: ☐ Visa ☐ Master Card ☐ American Express

☐ Payment enclosed ☐ Bill me later

Card # ____ Exp. date ____

Signature ____

Name ____

Address ____

City ____ State ____ Zip ____

ONLY $29.97! YOU SAVE OVER $5.00

Savings based on a full one-year cover price of $35.40. Canada & Mexico add $10 for surface mail per year. All other countries add $27 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc. **3463**

COMMENTS & SUGGESTIONS

November 1987, #133

Dear Reader,

Dr. Dobb's has a long tradition of listening to its readers. We like to hear when something really helps or, for that matter, bothers you. In this hectic world of ours, however, it is often difficult to take time to write a letter. This card provides you with an easy way to correspond and, if you include your name and address, we may use appropriate comments in The Letters column. Simply fill it out and drop it in the mail.

—Ed

Which articles or departments did you enjoy the most this month? Why?

____

____

____

Comments or suggestions ____

____

____

____

Name: ____

Address: ____

____

No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of
Software Tools
Box 3713
Escondidio, CA 92025-9843

No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of
Software Tools
Box 3713
Escondidio, CA 92025-9843

PLACE
STAMP
HERE

Dr. Dobb's Journal of
Software Tools
501 Galveston Drive
Redwood City, CA 94063

Dr. Dobb's Journal of Software Tools

The R_x for Programmers

Subscribe Now & Save Over 15% Off the Newsstand Price!

(continued from page 119)

```
sep 7 87
9-7
9/7
9/7/87
9 7 1987
9-7-1987
```

as are several other variants. Experiment and see if your favorite form works. If you use an asterisk in place of a month, then that day of every month is recognized—for example */9 will be recognized on the ninth of every month.

I've also added a non-Unix feature that recognizes days of the week (*monday, tuesday, wednesday,* and so forth). When these are used, memos are printed on the given day of every week (say, every Monday).

You can abbreviate month and day-of-the-week names by leaving off letters at the ends of the words, but you can't leave off so many letters that a conflict is created. For example, you can't use *Ju* because the program won't know if you mean June or July. The smallest possible abbreviations are shown in Listing One, page 108, on lines 42–51. Case is ignored, so *SEP, sep,* and *Sep* are all acceptable.

Calendar gets confused if you start the text portion of the line with a character that could be part of the date and a partial date is specified. For example, if you say:

```
9/17 4 for feast at Fran's
```

calendar will think that the date is September 19, 1904.

```
9/17/87 4 for . . .
9/17: 4 for . . .
```

are both OK, however; the first because the whole date is specified, and the second because the colon will separate off any trailing numbers. In general, avoid numbers, slashes, apostrophes, dashes, and periods at the beginning of the memo text.

The program itself is in Listing One. Most of it is in the *main()* subroutine, which starts on line 110. Today's date is fetched on lines 125 and 126, using the *time()* and *localtime()* functions described earlier. Word-to-month conversion is done in the *if* statement on lines 140–178. The program just looks at the first few characters in the word in the stupidest way possible. It didn't seem worth the effort to do things in a more efficient manner.

Numeric parts of the date are split off on lines 180–195. Note that the month is only modified (on line 188) if it hasn't been set previously (by the word-processing code). An asterisk is translated to the current month on line 182. The *stoi()* subroutine has appeared in this column several times in the past (it's not part of this month's listing). It works just like *atoi()* does, but it is passed a pointer to a character pointer and updates that pointer to point past the number.

The tests on lines 198–200 check for unspecified parts of the date and fill in the equivalent information for today if necessary. That is, if the year isn't specified, the current year is used and so forth.

The date is converted to an offset, in days, from the first of the year on lines 211–213. I'm doing this to make it easier to find tomorrow. Otherwise, the end of the month would present problems. The *offset_to_first_of_month* array is declared on lines 78–92. The *if* statement on line 212 adjusts for leap years (any year that is an even multiple of 4 except for even centuries—1900 wasn't a leap year). Finally, the appropriate lines are printed on lines 227–235.

Dateh.c

The dateh program is used to automatically create *#defines* for the current date and time so that you can put a visible date stamp into a program's log-in message. You can put it into a DOS batch file so that it's invoked automatically before calling make. If you're using my shell, you can do it with an alias:

```
alias make 'dateh; make'
```

The output file was described earlier. It's created only when a file called date.h already exists in the current directory and it was not modified today. It behaves in this way for two reasons. First, you don't

NEW!

Documentation is a PAIN!

. . . without DocuMotion™

We've found that well indexed and easily accessed documentation is a powerful tool and asset. Now you can simply pop up **DocuMotion** to access, display and print **your** documentation. **DocuMotion** builds indexed document libraries from documentation contained in your source code or any text file.

DocuMotion for programmers:
- Your documentation is available ANYWHERE, ANY TIME.
- Access, display and print your documentation by name or by user-defined categorization trees.
- 19 function Microsoft Windows-style menu bar.
- Powerful print & copy functions.

DocuMotion for project mgrs:
- Programmers produce more and better documentation.
- Reduced function redundancy.
- Greater programmer productivity.

DocuMotion for the PC:
- Runs memory resident or non-resident on any IBM PC/XT/AT or compatible.
- Compatible with all popular memory resident programs.
- Requires only 93K RAM.
- LAN compatible.

DocuMotion is for YOU:

Call NOW 1-612-884-5860

At a special introductory price of ONLY **$159.95** with ANSI 'C' document library.

Demo disk for $24.95 that puts the ANSI 'C' library functions on-line.

NWP - Intelligent Solutions, Inc. P.O. Box 20478 Bloomington, MN. 55420-0478

CIRCLE 400 ON READER SERVICE CARD

UNLEASH YOUR 80386!

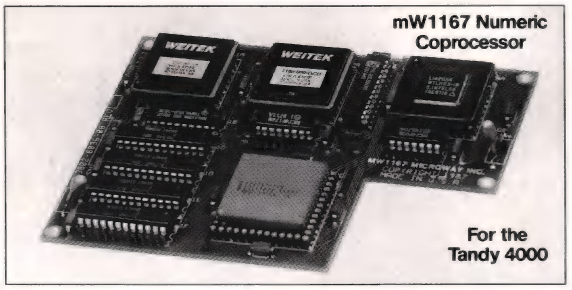
Your 80386-based PC runs at least twice as fast as your old AT. This is good, but not great. The products described below will unleash the true potential of your 80386, giving you 4 to 16 times the power of your old AT. These new MicroWay products include a family of 80386 native code compilers and the mW1167 numeric coprocessor.

Examples of the increases in capacity and performance include:

- Programs compiled with MicroWay NDP Fortran-386 execute 2 to 8 times faster than those compiled with existing 16-bit Fortrans. NDP Fortran-386 can also address up to 4 gigabytes of memory instead of the standard 640 kbytes. MicroWay's NDP compilers and the programs they generate run on MS-DOS or Unix V.
- NDP Fortran-386 generates code for the 80287, 80387 or MicroWay's mW1167. The mW1167 has a floating point throughput exceeding 2.5 megaflops, which is 4 to 5 times the throughput of an 80387 and is comparable to the speed achieved by the VAX 8600.

Equally important, whichever MicroWay product you choose, you can be assured of the same excellent pre- and post-sales support that has made MicroWay the world leader in PC numerics and high performance PC upgrades. For more information, please call the Technical Support Department at

617-746-7341



MicroWay® 80386 Support

MicroWay 80386 Compilers

NDP Fortran-386 and **NDP C-386** are globally optimizing 80386 native code compilers that support a number of Numeric Data Processors, including the 80287, 80387 and mW1167. They generate mainframe quality optimized code and are syntactically and operationally compatible to the Berkeley 4.2 Unix f77 and PCC compilers. MS-DOS specific extensions have been added where necessary to make it easy to port programs written with Microsoft C or Fortran and R/M Fortran.

The compilers are presently available in two formats: Microport Unix 5.3 or MS-DOS as extended by the Phar Lap Tools. MicroWay will port them to other 80386 operating systems such as OS/2 as the need arises and as 80386 versions become available.

The key to addressing more than 640 kbytes is the use of 32-bit integers to address arrays. NDP Fortran-386 generates 32-bit code which executes 3 to 8 times faster than the current generation of 16-bit compilers. There are three elements each of which contributes a factor of 2 to this speed increase: very efficient use of 80386 registers to store 32-bit entities, the use of inline 32-bit arithmetic instead of library calls, and a doubling in the effective utilization of the system data bus.

An example of the benefit of excellent code is a 32-bit matrix multiply. In this benchmark an NDP Fortran-386 program is run against the same program compiled with a 16-bit Fortran. Both programs were run on the same 80386 system. However, the 32-bit code ran 7.5 times faster than the 16-bit code, and 58.5 times faster than the 16-bit code executing on an IBM PC.

NDP FORTRAN-386™	**$595**
NDP C-386™	**$595**

MicroWay Numerics

The **mW1167™** is a MicroWay designed high speed numeric coprocessor that works with the 80386. It plugs into a 121 pin "Weitek" socket that is actually a super set of the 80387. This socket is available on a number of motherboards and accelerators including the AT&T 6386, Tandy 4000 and MicroWay Number Smasher 386 (Jan. '88). It combines the 64-bit Weitek 1163/64 floating point multiplier/adder with a Weitek/Intel designed "glue chip". The mW1167™ runs at 3.6 MegaWhetstones (compiled with NDP Fortran-386) which is a factor of 16 faster than an AT and 3 to 5 times faster than an 80387 .... **$1495**

Monoputer™ - The INMOS T800-20 Transputer is a 32-bit computer on a chip that features a built-in floating point coprocessor. The T800 can be used to build arbitrarily large parallel processing machines. The Monoputer comes with either the 20 MHz T800 or the T414 (a T800 without the NDP) and includes 2 megabytes of processor memory. Four or more Transputers can be easily linked together to form a Quadputer. A single T800 is comparable in speed with an mW1167-equipped 80386. The compilers to drive one or more Monoputers include Occam, C, Fortran, Pascal and Prolog.

Monoputer T414-20[1]	**$1495**
Monoputer T800-20[1]	**$1995**
Biputer™ T800/T414[2]	**$4995**
Quadputer™ T414-20[2]	**$6995**

[1] Includes Occam [2] Includes TDS

80287 ACCELERATORS

287Turbo-10	$450
287Turbo-12	$550
287TurboPlus-12	$629

80386 Multi-User Solutions

AT8™ – This intelligent serial controller is designed to handle 8 users (16 with two boards) in a Xenix or Unix environment with as little as 3% degradation in speed. It has been tested and approved by Compaq, Intel, NCR, Zenith, and the Department of Defense for use in high performance 80286 and 80386 Xenix or Unix based multi-user systems **$1299**

MicroPort Unix 5.3 is a port of the new Unix 5.3 to the 80386. MicroWay NDP-386 compilers currently run on this version of UNIX.
MicroPort Unix 5.3 .... from $399

PC-MOS-386™ is an 80386 operating environment that turns an AT with an AT8 into an MS-DOS multi-user system. The system makes it possible to run applications such as Lotus 1-2-3 on terminals. The operating system also has a Phar Lap compatibility mode that runs programs developed with the Phar Lap versions of MicroWay's compilers .... **from $199**

Phar Lap™ created the first tools that make it possible to develop 80386 applications which run under MS-DOS yet take advantage of the full power of the 80386. These include an 80386 monitor/loader that runs the 80386 in protected linear address mode, an assembler, linker and debugger. These tools are required for the MS-DOS version of the MicroWay NDP Compilers.
Phar Lap Tools...................... $399

MATH COPROCESSORS

80387-16 16 MHz	**$495**
80287-10 10 MHz	**$349**
80287-8 8 MHz	**$259**
80287-6 6 MHz	**$179**
8087-2 8 MHz	**$154**
8087 5 MHz	**$99**

Micro Way

The World Leader in PC Numerics

P.O. Box 79, Kingston, Mass. 02364 USA (617) 746-7341
32 High St., Kingston-Upon-Thames, U.K., 01-541-5466

CIRCLE 300 ON READER SERVICE CARD

C CHEST
(continued from page 121)

want myriad date.h files littering all your directories. Second, if you're using a make utility, you only want to recompile the file that *#includes* date.h once a day.

The program's behavior can be modified with the following command-line switches:

-f—forces an update, even if the date stamp is today's.
-v—verbose operation. The program tells you about the current date and so forth as it runs.
-c—forces dateh to create the file if it doesn't already exist.

The source code for dateh is in Listing Two, page 110. It is much more straightforward than calendar.c. Command-line switches are processed on lines 50–65. The *stat()* call on line 69 does two things. It tests for the existence of a file called date.h (*stat* returns –1 if the file doesn't exist), and if the file does indeed exist, it initializes the *stats* structure with the relevant statistics. Here, I'm interested in the last-modified date stamp held in the *st_mtime* field. This field is converted to a *tm* structure by the *localtime()* call on line 79. I do the same thing for today's date and time on line 86. Finally, the two times are compared on line 91. The rest of the subroutine just modifies the date.h file if the date stamps don't match, getting the relevant information from the *tm* structure pointed to by *t*.

Note

1. My shell is described in my book *On Command: Writing a Unix-Like Shell for MS-DOS* (Redwood City, Calif.: M&T Books, 1987).

DDJ

(Listings begin on page 108.)

Vote for your favorite feature/article.
Circle Reader Service **No. 7.**

NEW!
VERY FAST EGA TOOLKIT
FOR TURBO C® and MICROSOFT® C

For high-resolution (640x350, 16/64) color graphics, this toolkit is unsurpassed. 90 well-integrated functions, incorporating design features hitherto not implemented in microcomputers, crafted to achieve highest speed possible in mode 16. Requires only 128k-EGA board. 384k+ system. Small, compact, medium and large models.

*Probably fastest spline functions available for microcomputer.
*Fast running fills (including spline-and-fill), versatile flood.
*Create graphics fonts (any size), or use HP LASERJET PLUS™ fonts.
*Set "text window" for auto-scroll/wrap (optional word-wrap): works with both IBM® (25/43) and graphics fonts.
*Separate viewport clips geometric figures and runfill/flood areas.
*Over 12 get/put/mov/swap/mask/paint/save/load 'pic' functions to achieve maximum flexibility/speed in animation of high-resolution, 16-color images.
*Includes source for MS MOUSE functions similar to those used by popular PAINT programs.
*Helpful demo-tutorial shows sample code and its effects.

INTRODUCTORY OFFER: ONLY $49

Prototype Systems Ltd.
637 17th Street
Boulder, CO 80302
1-800-628-2828 ext. 493

'Faster and better in no time'

Create better, faster, higher quality and easier to read progams in a fraction of the time. Let your system do the work for you. With 23 powerful, state of the art tools for the IBM PC and compatibles, both beginners and experts will find programming a breeze.

- Powerful, common user interace
- Interactive and batch execution
- Online documentation
- Self-explanatory error messages

C Programmer's Toolbox
Volumes I & II

- Monitor program/system execution
- Beautify program listings
- Determine program flow/critical path
- Trace/verify variable usage
- And more....

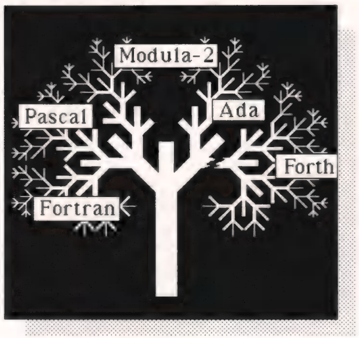
At $79.95 per volume or $130 for both with a **30 day trial**, the Toolbox is simply the **best value** today. In addition, the documentation is packed with examples and tips on creating better programs.

Why waste time, **call or write us today.**

TM
MMC AD Systems
Box 360845 Milpitas, California 95035
(408) 263-0781

CIRCLE 192 ON READER SERVICE CARD

Heuristic Searching



by Namir Clement Shammas

This month I'm going report on some new algorithms for heuristic searching. Heuristic approaches, as you may know, differ from conventional searching by improving their performance over time, by "learning" the characteristics of the data and the requests normally made of it. In this column I'll present algorithms for searching both unsorted and sorted arrays: the methods for searching through unsorted arrays borrow from statistical time-series analysis to create a short-term heuristic memory, whereas the methods for sorted arrays are variations on the traditional index table search scheme.

Unsorted Arrays

The simplest case of a heuristic search involves searching through an unsorted array or list. The search begins with the first array element and examines the rest of the array until a match is found. If the matched element is not the first array member, it is swapped with the previous array element. Thus, the most frequently sought elements tend to bubble toward the front end of the array.

Listing One, page 112, shows a Pascal function that implements a version of the simple heuristic search algorithm. The Pascal function passes an array of data as well as an array of indices. The values of the indices rather than the members of array *Data* are swapped. This approach is more convenient and faster when record structures are used and the search is performed for various keys.

This simple method may be too simple, though. Using it, you may experience any of the following situations, based on the pattern of the data and requests:

1. The statistical probability distribution for seeking any element is uniform. In this case, the benefit of using this method over a simple linear search depends on the actual sequence of sought elements. Thus, the advantage of a heuristic search, at least of this simple kind, is most likely minimal.
2. The statistical probability distribution for seeking array elements strongly favors a particular set of data. Using the simple heuristic method is the most efficient in this situation.
3. There is a shifting trend for seeking different data sets in the array (that is, the statistical probability distribution for seeking any element is also a function of time or call sequence). Here the efficiency of the algorithm is also shifting: after a specific set of data bubbles toward the front of the array, the trend shifts in favor of another set. This causes the members of the new set to bubble toward the front and displace the ones from the previous set. As a result, the efficiency of the algorithm drops to a minimum during a shift in such trends because the sought elements are either in the middle or at the tail.

It is this last case that I want to examine more closely. It is possible to improve the algorithm to take advantage of any patterning of need if you consider the following questions: When do you start searching at the lower array bound? When do you start backward searching at the upper array bound? And when do you start bidirectional searching beginning at the median element?

These three questions have a common solution: you predict the location of the next element from the history of search locations. There exists in statistical time-series analysis a tool to do just this. In the following discussion, I use the weighted average method to provide projections for the next sought location. The equation (which I'll call equation 1) involved is:

$$L[n+1] = (1-f)(L[n] + fL[n-1] + f^2L[n-2] + \ldots)$$

where L[1] . . . L[n] is the array of locations found and f is a fractional factor. L[n] is the location of the last search, L[n–1] is the location of the one before it, and so on. The value of f is usually calculated from the autocorrelation coefficient associated with a given set of values of array L. For the sake of reducing computing time, however, f is assigned a fixed value (say 0.70). As the value of the f factor approaches unity, more weight is given to the "older" data, and vice versa.

With the predicted next location at hand, you can determine the search scheme: first to last, bidirectional to middle, and last to first. This is expressed in the following conditions:

1. If $L[n+1]/(\text{array size}) > 0.65$, then use the last-to-first search scheme.
2. If $L[n+1]/(\text{array size}) < 0.35$, then use the first-to-last search scheme.
3. If $0.35 <= L[n+1]/(\text{array size}) <= 0.65$, then use the bidirectional-middle search scheme.

Listing Two, page 112, shows the Pascal function *Search2*, which employs the modified heuristic method just described. It uses equation 1 with f = 0.70. The last four search location values are used to predict the next search location, and the

Avoid extra steps.

You've got better things to do than repeat the same steps. Over . . . and over . . . and over. Up your productivity with Greenleaf Software.

With more than 70 new functions added to our popular libraries, Greenleaf is now the most complete and mature C language function resource available. It's no wonder we've been rated the best. Winning program developers in major corporations such as IBM, EDS and GM have proven our reliability in thousands of applications.

Step Lively

New Greenleaf Functions v.3.10 includes 295 of the functions you've been asking for — DOS, disk, video, color text and graphics, string, time/date, keyboard, plus many more! With Greenleaf, you'll finish faster.

Cut Corners

When it comes to merging information, the new Greenleaf Comm Library v.2.10 is the fastest communications facility of its kind. Over 120 functions — ring buffered, interrupt-driven asynchronous communications. And, only Greenleaf gives you the power to build a 16-port communication system.

Get on the Fast Track

Order your new Greenleaf library today! See your dealer or call 1-800-523-9830.

Greenleaf Comm Library	$185.00
Greenleaf Functions	$185.00
Greenleaf DataWindows	$225.00
Greenleaf C Sampler	$ 94.50
Digiboard Comm4	$325.00
Digiboard Comm8	$535.00

In stock, shipped next day.

Greenleaf DataWindows and Turbo C

DataWindows, the finest C programming windows tool available, puts windows, transaction data entry and menus at your fingertips.

Our new TURBO C versions are ready to get you going fast! And, our new 3-in-1 C Sampler for only $94.50 supports both Turbo C and Quick C with comm, windows, menus and more! Our libraries support all popular C compilers for MS DOS.

GREENLEAF
Software

Call Toll Free:

800-523-9830

In Texas and Alaska:

214-446-8641

Greenleaf Software, Inc.
16479 Dallas Parkway, Suite 570
Dallas, Texas 75248

CIRCLE 97 ON READER SERVICE CARD

(continued from page 124)

predicted next location is calculated as a fraction of the total number of array elements. This fractional value is then used in selecting the search scheme. Notice that the function updates the search location history only when matches are found.

Ordered Arrays

Conventional (nonheuristic) searching through ordered arrays usually employs index tables. The search for a given datum begins in the index table whose entries define a range to be examined in the array. The sought datum may or may not find its match within that range and is guaranteed not to match outside the range. Normally, the index tables are specified ahead or initialized to set the table entries based on the available data. For an array in which no new members are added and none are removed, the index table remains the same.

This is the basis on which I will now develop and discuss two heuristic search methods for ordered arrays. The basic scheme I use is the dynamic modification of the index table as influenced by the sought data.

In the first of these methods, the index table is constantly changing at run time. It is first initialized by taking evenly spaced elements from the data array. The index table itself is an array with a lower bound of 0. The value of the table entry at the lower bound points to the first element (with common values of 0 or 1) of the data array. The basic algorithm for searching and updating the heuristic index table is:

1. Scan the array elements to compare the search datum with the table entries. Determine the indices to the "first" and "last" members of the data array. Let K be the table index that points to the first member.
2. Search the data array in the range first . . . last.
3. Return the index of the matching element and replace the Kth index table entry with the matching data if a match is found.
4. Return a special coded integer if no match is found.

Listing Three, page 112, shows a Pascal function *Search3* that implements this algorithm and a procedure that initializes the index table.

One of the weaknesses of this heuristic search method is that the index table may contain entries of clustered data. The loss of uniformly distributed or well-spread indices is a major factor in the deterioration of search efficiency. To remedy this clustering problem, I suggest a second method. This new variant combines a few aspects from traditional index tables with the ones in the previous method. The result is a heuristic index table with both fixed and replaceable entries. The index table consists of several sections such that each section begins with one fixed entry followed by m replaceable entries. The algorithm is very similar to that of the previous method:

1. Search the data array in the range first . . . last.
2. If a match is found, return the index of the matching element.
3. If (K–1) is not 0 or a multiple of (m+1), replace the Kth index table entry with the matching data.
4. If no match is found, return a special coded integer.

The test in step 3 protects the fixed entries from being overwritten. This simple modification provides anchor points in the heuristic index table. You can set up the index table using the same method as for the previous search algorithm.

I've discussed all the algorithms I promised, but I can't resist pointing out an interesting variation of the second search method that can be applied to data stored in binary trees. In this variation, a modified index table employs pointers instead of integer indices, with the pointers used to indicate the node at which you start the search in a binary tree rather than taking the traditional approach of starting at its root node. By using the combination of fixed and replaceable pointers in the index table, efficient searching is maintained. Because the fixed array is stored in a binary tree, you can easily construct a balanced or near-balanced binary tree, which adds even more to the efficiency of the search.

If you have developed other heuristic search schemes, I would like to hear from you. Please write to me care of *DDJ*.

Availability

All the source code for articles in this issue is available on a single disk. To order, send $14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call (415) 366-3600, ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

DDJ

(Listings begin on page 112.)

Vote for your favorite feature/article.
Circle Reader Service **No. 8.**

Turbo Tech Report Speaks Your Language.

Turbo Pascal Articles and Reviews

News and commentary

A disk filled with Turbo Pascal code!

TURBO TECH REPORT | The Independent Newsletter for Turbo Pascal Programmers

Volume 1, No. 1 September/October 1986

Turbo Watch

Editorial by Namir Clement Shammas

Welcome to the premiere issue of Turbo Tech Report; we hope you find it a useful tool and we look forward to your feedback. You can look forward to Turbo Pascal market news; reviews of libraries, utilities, and books; in-depth articles; and source code on disk in every issue.

When the phone rang, I picked it up quickly: details of Borland's just-announced Turbo Pascal for the Macintosh. I asked the obvious questions. Did the implementation take advantage of the Mac user interface? And if it did, what about the vast number of existing Turbo Pascal programs not written with the Mac interface in mind? Did it offer any clues about the nature of the next MS-DOS version? Was it compatible with the Mac Hierarchical File System (HFS)? When would it be released? How much?

Turbo Tech Report will make a point to address the entire Turbo Pascal programming community, by discussing the issue of porting code across the three Turbo Pascal environments.

As I keyed in the answers, Paul McCartney was crooning in the background "open the door and let them in." It's true that TPMac will let the Macintosh Pascal programmer in on the world of Turbo Pascal programming, and that's good: the Mac has suffered from the lack of a quick-start software development environment ever since it was released. Even Apple recognized the importance of TPMac and sent chairman John Sculley to the product announcement press conference. The announcement is of importance to the MS-DOS crowd of Turbo programmers as well; they're getting a sneak preview of some features that are sure to be included in the MS-DOS Turbo Pascal Version 4.0.

But the real significance of this product is in what it offers to the Turbo Pascal programming community -- Borland has made it possible for a number of existing Turbo Pascal programs to run without modification market for Turbo Pascal programs and utilities has now expanded to all the major personal computer operating system environments, from CP/M through MS-DOS to the Mac's new HFS.

The Three Turbo Pascal Environments

CP/M | IBM PC Compatibles | Macintosh

Turbo Tech Report will take pains to address this entire Turbo Pascal programming community by supplying portable code whenever feasible. In any case, we'll make a point to address the entire Turbo Pascal programming community, by discussing the

Continued on page 15

IN THIS ISSUE:

Mathematical Expression Parsers for Run-Time Evaluation By Namir Clement Shammas	2
DDAY: A Turbo Pascal Deadline Calculator By Morris Simon	9
Turbo Power Tools Plus from Blaise Computing Reviewed by Vernon Stant	11
TDebugPLUS from TurboPower Software Reviewed by Namir Clement Shammas	12
Turbo Extender from TurboPower Software Reviewed by Vernon Stant	13
Programmer's Utilities from TurboPower Software Reviewed by Vernon Stant	14
Turbo Pascal Tricks and Tips Reviewed by Michael Swaine	16
Mastering Turbo Pascal Reviewed by Nick Spain	17
News Bits	18
On Disk	19

Turbo Tech Report 501 Galveston Dr. Redwood City, CA 94063

The newsletter/disk publication for Turbo Pascal® users

Are you looking for powerful utilities written in Turbo Pascal that you can use to develop software or incorporate into your programs? Are you interested in improving and expanding your Turbo Pascal programming skills?

Then you deserve a subscription to *Turbo Tech Report*, the bimonthly newsletter/disk publication from the publishers of *Dr. Dobb's Journal* and *Micro/Systems Journal*. Each issue delivers more than 250K of Turbo Pascal source code programs on disk, and 24+ pages of articles, Turbo Pascal software and book reviews, and analysis and commentary. It's the only publication delivering such focused technical articles with code on disk. Each valuable issue contains:

- ***Articles*** on topics like speedy 3D graphics, mathematical expression parsers, creating global gotos, memory resident and AI applications and more—all written by Turbo experts.
- ***Reviews*** of the latest Turbo Pascal software programs from companies like Borland International, Blaise Computing, Media Cybernetics, Nostradamus, and more!
- ***News and commentary*** detailing the latest products and developments in the Turbo Pascal programming community.
- ***A disk filled with Turbo Pascal code!*** You'll get the Turbo Pascal utilities and routines discussed in the newsletter's articles, as well as applications developed by Turbo users from around the world. You'll receive programs that make labels, generate menus, provide faster screen access, transfer files, and more!

If you're an expert Turbo Pascal programmer or a novice interested in expanding your Turbo skills, you need a publication that speaks your language: *Turbo Tech Report*. Subscribe today at the special price of just $99—that's 33% off the regular price of $150. To order by credit card, call toll-free 1-800-533-4372. Or mail the coupon with your payment to *Turbo Tech Report*, 501 Galveston Drive, Redwood City, CA 94063.

—**Yes!** I want a one-year subscription to *Turbo Tech Report* (6 issues with 6 disks) for $99.

Format: ☐ PC/MS-DOS ☐ Macintosh

CP/M: ☐ Kaypro ☐ Osborne ☐ Apple

PAYMENT MUST ACCOMPANY ALL ORDERS

☐ Check/money order enclosed.

☐ Charge my: ___ VISA ___ M/C ___ AmExp.

Card # ____________________ Exp. ________

Signature ____________________

Name ____________________

Address ____________________

City ____________ State ______ Zip ________

Orders outside U.S.: add $30.

CIRCLE 119 ON READER SERVICE CARD

THE PROGRAMMER'S SHOP

helps save time, money and cut frustrations. Compare, evaluate, and find products.

RECENT DISCOVERY

OS/286 & 386 - Access 16M RAM in 286, 386 native mode. DOS interface, access. Use existing compilers or 32-bit FORTRAN, C, PASCAL. PC $ 495

AI-Expert System Dev't

Arity Combination Package	PC	$ 979
System - use with C	MS	$ 229
Auto-Intelligence	PC	$ 739
CxPERT - shell for C	MS	$ 259
Experteach II	PC	$ 339
Exsys	PC	$ 289
Level 5	MS	$ 659
Intelligence/Compiler	PC	$ 739
T.I.: PC Easy	PC	$ 435
Personal Consultant Plus	PC	$2589
Turbo Expert-Startup(400 rules)	PC	$ 129
Corporate (4000 rules)	PC	$ 359

AI-Lisp

PC Scheme LISP - by TI	PC	$ 85
Star Sapphire	MS	$ 459
TransLISP - learn fast	MS	$ 79
TransLISP PLUS	MS	$ 149

Others: IQ LISP ($155), IQC LISP ($269)

AI Prolog

APT - Active Prolog Tutor - build applications interactively	PC	$ 49
ARITY Prolog - Interpreter	PC	$ 229
COMPILER/Interpreter-EXE	PC	$ 569
MicroProlog Prof. Comp./Inter.	MS	$ 439
Prolog-86 - Learn Fast	MS	$ 89
Prolog-86 Plus - Develop	MS	$ 199
TURBO PROLOG by Borland	PC	$ 69
Turbo Prolog Toolbox	PC	$ 69

Basic

BAS_C - economy	MS	$ 179
BAS_PAS - economy	MS	$ 135
Basic Development System	PC	$ 105
Basic Development Tools	PC	$ 89
BetterBASIC	PC	$ 129
Exim Toolkit - full	PC	$ 45
Finally - by Komputerwerks	PC	$ 85
Inside Track	PC	$ 49
Mach 2 by MicroHelp	PC	$ 55
Peeks 'n Pokes	PC	$ 35
QBase Screens	MS	$ 79
QuickBASIC	PC	$ **69**
Quick Pak - by Crescent Software	PC	$ 59
Quick-Tools by BC Associates	PC	$ 109
True Basic	PC	$ 79
Turbo BASIC - by Borland	PC	$ 69
Turbo BASIC Database Toolbox	MS	$ 69

FEATURES

XENIX 386 Toolkit by Santa Cruz. Tools & OS kernel support 4 Gigabyte address space, demand paging, virtual memory paging. Includes MS C, MASM, debugger, file utilities, link kit, runtime library. PC $ 379

HI-SCREEN XL - unlimited overlapping windows, pulldown, popup, Lotus menus, on-line help support. Field checking, scrolling. Screen capture, more. Multi-language support. PC $ 129

Note: Mention this ad. Some prices are specials. Ask about COD and POs. Formats: 3" laptop now available, plus 200 others. UPS surface shipping add $3/item. All prices subject to change without notice.

"FREE" Software

Instant Programmer's Help for MSDOS developers by Second City Software. Memory-resident (TSR) utility lets you access valuable programmer's reference material from within any program. OS services, BIOS calls, ASCII codes, line and box drawing characters, video modes, keyboard scan codes, interrupts, 8088 assembly language, memory maps, FCB.

Normally $79. With any order over $250 by 11/30/87 get **Instant Programmer's Help** (with 2-month life span) **FREE.**

Our Services:

- Programmer's Referral List
- Compare Products
- Help find a Publisher
- Evaluation Literature FREE
- BBS-7 PM to 7 AM 617-740-2611
- Dealers Inquire
- Newsletter
- Rush Order
- Over 700 products
- National Accounts Center

Cobol

Microfocus Cobol I/Q	PC	$ 395
Cobol spII	PC	$ 329
Level II COBOL	MS	$ 279
Personal COBOL	PC	$ 119
Professional Cobol	PC	$1195
Microsoft COBOL	MS	$ 429
COBOL Tools	PC	$ 209
Realia - very fast	MS	$ 895
PCDT by Procode	MS	$ 949
Ryan McFarland COBOL	MS	Call
COBOL-85	MS	Call
Screenplay by Flexus	PC	$ 119

C Language-Compilers

AZTEC C86 - Commercial	PC	$499
C86 PLUS - by CI	MS	$359
Datalight Optimum-C	MS	$ 99
with Light Tools by Blaise	PC	$168
Lattice C - from Lattice	MS	$269
Let's C Combo Pack	PC	$ 99
Microsoft C 5.0- Codeview	MS	$**275**
Microsoft Quick C	MS	$ 69
Rex - C/86 - standalone ROM	MS	$695
Turbo C by Borland	PC	$ 69

C Libraries-Communications

Asynch by Blaise	PC	$ 125
Essential Comm Library	PC	$ 125
With Breakout Debugger	PC	$ 189
Greenleaf Comm Library	PC	$ 129
Multi-Comm - add multitasking	PC	$ 135

DBASE Language

Clipper compiler	PC	$ 399
dBASE II	MS	$ 329
dBase III Plus	PC	$ 429
dBASE III LANPack	PC	$ 649
DBXL Interpreter	PC	$ 139
FoxBASE+ - V2.0	MS	$ 289
Quicksilver by Word Tech	PC	$ 499

DBASE Support

dAnalyst	PC	$ 89
dBase Tools for C	PC	$ 65
dBrief with Brief - Auto-Indent, views structures	PC	Call
dBC ISAM by Lattice	MS	$ 169
Documentor - dFlow superset	MS	$ 229
Genifer by Bytel-code generator	MS	$ 279
QuickCode III Plus	MS	$ 239
R&R Report Generator	MS	$ 139
Seek-It - Query-by-example	PC	$ 79
Silver Comm Library	MS	$ 139
Tom Rettig's Library	PC	$ 79
UI Programmer-user interfaces	PC	$ 249

RECENT DISCOVERY

The Baler - compiles Lotus 1-2-3 (1A, 2.01, or VP-Planner). Protect code, speed execution, save disk space, password capability, user does not need 1-2-3. **No Royalties.** PC $ 459

Editors for Programming

BRIEF Programmer's Editor	PC	Call
EMACS by UniPress Source: $895		$ 265
Epsilon - like EMACS	PC	$ 149
KEDIT - like XEDIT	PC	$ 99
ME Macro Editor - Source	PC	$ 79
Micro Focus Micro/SPF	PC	$ 139
PC/EDT - macros	PC	$ 229
PC/VI - by Custom Software	MS	$ 109
Personal REXX - V1.6	PC	$ 99
PMATE - power, multitask	PC	$ 109
SPF/PC - Version 2.0	PC	$ 189
Vedit	MS	$ 99
Vedit PLUS	MS	$ 129

C Language-Interpreters

C-terp by Gimpel - full K & R	MS	$219
C Trainer - by Catalytix	PC	$ 89
INSTANT C - Source debug, Edit to Run-3 seconds, .OBJs	MS	$369
Interactive C by IMPACC Assoc.	PC	$189
Run/C Professional	MS	$155
Run/C Lite	MS	$ 79

C Libraries-General

Blackstar C Function Library	PC	$119
C Essentials - 200 functions	PC	$ 75
C Tools Plus (1 & 2) - Blaise	PC	$135
C Utilities by Essential	PC	$119
Entelekon C Function Library	PC	$119
Entelekon Superfonts for C	PC	$ 45
Greenleaf Functions-portable, ASM		$139
LIGHT TOOLS by Blaise	PC	$ 69

C Libraries-Files

C Index by Trio	MS	$ 89
/File is object only	MS	$ 89
/Plus is full source	MS	$319
BTree by Soft Focus	MS	$ 69
CBTREE - Source, no royalties	MS	$ 99
CTree by Faircom - no royalties	MS	$315
ctree w/ rtree	MS	$519
rtree - report generation	PC	$239
dB2C Toolkit V2.0	MS	$249
dbQUERY - ad hoc, SQL - based	MS	Call
dbVISTA - full indexing, plus optional record types, pointers, Network.		
Object only - MS C, LAT,	C86	Call
Source - Single user	MS	Call
Source - **Multiuser**	MS	Call
dBx - translator	MS	$299
w/source to library	MS	$429

FEATURES

HUMMINGBOARD 386-Develop 2.6 or 7.9 times faster than a 8 MHZ AT. AT or XT addin board uses dual processors for Speed and Hardware Debugging. 16 MHZ or 20 MHZ. Call about Benchmarks, Trial Program.

Greenleaf C Sampler - 107 functions for communications, windows, menus, business graphics, keyboard control from popular Greenleaf libraries. Turbo Quick C support. No royalties. PC $169

We support MSDOS (not just compatibles), PCDOS, Xenix-86, CPM-80, Macintosh, Atari ST, and Amiga.

CIRCLE 133 ON READER SERVICE CARD

THE PROGRAMMER'S SHOP

provides complete information, advice, guarantees and every product for Microcomputer Programming.

C Support-Systems

Advantage C + +	PC	$479
C Sharp - **realtime,** tasks.	PC	$495
C ToolSet - DIFF, xref, source	MS	$ 89
Multi-C - multitasking	PC	$135
PC LINT-Checker. Amiga $89	MS	$ 99
Pfantasy Pac - by Phoenix	PC	$599
PforCe + +	PC	$215
Pre-C - Lint-Like	MS	$155
Quickshell - script compiler	PC	$349
Sapiens V8 - 8M workspace	PC	$269
SECURITY LIB - Source $229	PC	$115
Timeslicer	PC	$265
with library source	PC	$895

C-Screens, Windows, Graphics

C Power Windows by Entelekon	PC	$109
C Worthy Interface Library	PC	$249
Curses by Aspen Scientific	PC	$109
dBASE Graphics for C	PC	$ 69
ESSENTIAL GRAPHICS - fast	PC	$185
FontWINDOW/PLUS	PC	$229
GraphiC - new color version	PC	$279
Greenleaf Data Windows	PC	$159
w/source	PC	$269
LightWINDOWS/C-for Datalight C	PC	$ 79
Screen Ace Form Master	PC	$195
TurboWINDOW/C - for Turbo C	PC	$ 79
Vitamin C - screen I/O	PC	$159
Windows for C - fast	PC	$189
Windows for Data - validation	PC	$319
View Manager - by Blaise	PC	$199
ZView - screen generator	MS	$129

Fortran & Supporting

50:More FORTRAN	PC	$ 95
ACS Time Series	MS	$389
I/O Pro - screen development	PC	$129
MACFortran by Microsoft	MAC	$229
MS Fortran - 4.0, full 77'	MS	$279
No Limit - Fortran Scientific	PC	$109
Numerical Analyst by Magus	PC	$249
RM/Fortran	MS	Call
Scientific Subroutines - Matrix	MS	$129

Multilanguage Support

BTRIEVE ISAM	MS	$185
BTRIEVE/N-multiuser	MS	$455
Flash-Up Windows	PC	$ 79
GSS Graphics Dev't Toolkit	PC	$375
Hoops Graphics Library	PC	$549
Informix 4GL-application builder	PC	$789
Informix SQL - ANSI standard	PC	$639
NET-TOOLS - NET-BIOS	PC	$129
Opt Tech Sort - sort, merge	MS	$ 99
PANEL	MS	$215
Panel Plus	MS	$395
Pfinish - by Phoenix	MS	$229
PolyBoost - speed I/O, keyboard	PC	Call
Polylibrarian by Polytron	MS	Call
PVCS Corporate or Personal	MS	Call
Report Option - for Xtrieve	MS	$109
Screen Sculptor	PC	$ 89
SSP/PC - 145 + math routines	PC	$269
Synergy - create user interfaces	MS	$375
Xtrieve - organize database	MS	$199
ZAP Communications - VT 100	PC	$ 89

RECENT DISCOVERY

ACTOR by Whitewater Group - Graphic, object-oriented language for prototyping, interactive programs and MS Windows applications. Fast feedback, integrated editor, debugger. Call to multiple languages. PC $ 419

Other Languages

APL*PLUS/PC	**PC**	**$ 369**
CCS Mumps - Multiuser	PC	$ 359
CCS Mumps - Single user	PC	$ 49
Microsoft MASM	MS	$ 98
Modula-2 - V3.0 Compiler Pack	PC	$ 89
Modula-2 - V3.0 Dev. System	PC	$ 219
PC Forth + - by Lab Micro	PC	$ 189
Smalltalk/V	MS	$ 85
SNOBOL4 + - great for strings	MS	$ 80
UR/Forth	MS	$ 259

Debuggers

386 Debug - by Phar Lap	PC	$129
Breakout - by Essential	PC	$ 89
CODESMITH - visual	PC	$ 99
C SPRITE - data structures	PC	$119
DSD87	PC	$ 89
Periscope I	PC	$289
Periscope II	PC	$139
Periscope II-X	PC	$105
Periscope III - 10 MHZ Version	PC	$899
Pfix-86 Plus - by Phoenix	PC	$225
Turbo TDebug	PC	$ 49
SoftProbe II - embedded systems	PC	$695

Call for FREE Report: Interviews with Authors and Users of Products Below

Now you see it; Now you don't!

TSR's made easy with /*resident_C*/

Finally! You write your C program and we make it resident. No strings attached. Want to process interrupts also? No problem. **/*resident_C*/** is a set of C functions that enable you to process interrupts and/or make your programs "terminate stay resident" like Sidekick.

We've done the research, testing, and grunt work to make your TSR programs safe, compatible and easy. Be aware of the "ifs, ands and buts" when writing interrupt handling software. Only **/*resident_C*/** has all that you need to write reliable interrupt handlers without the worries.

We don't ask you to trust us either. Our manual explains what we are doing very clearly and our documented source code is available. In addition, our working demos give you clear examples of resident programs, interrupt handlers and resident libraries. There is no other product that can do what **/*resident_C*/** can do for you.

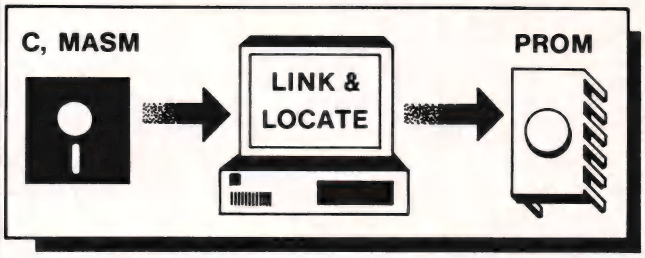
/* resident_C */ is the perfect complement to the other library products available from Essential Software.

ESI

List: $99 / with Source add $99
PS: $79 / with Source add $79

CIRCLE 301 ON READER SERVICE CARD

FIRMWARE DEVELOPMENT



LINK & LOCATE enables PC users to produce ROM-based firmware for 8086/87/186 from object files generated by popular C compilers, such as from Microsoft, Lattice and Borland's Turbo C, and MASM from Microsoft, Provides full control of segment placement anywhere in memory. Supports output of Intel HEX file for PROM programmers, Intel OMF absolute object file for symbolic debuggers and in-circuit emulators. Includes Intel compatible linker, locator, librarian, hex formatter and cross reference generator. $350.

NEW! Includes utility to support PC based PROM programmers.

SYSTEMS & SOFTWARE INC.

PS Price: $315

3303 Harbor Blvd., C11, Costa Mesa, CA 92626
Phone (714) 241-8650 FAX (714)241-0377 TWX 910-695-0125

CIRCLE 302 ON READER SERVICE CARD

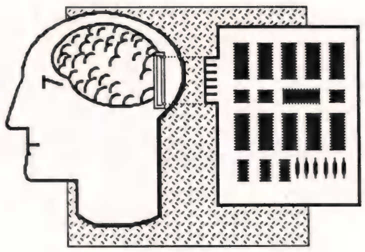
Call or Circle Reader Service Number for Your FREE Study.

HOURS: 8:30 A.M.-8:00 P.M. E.S.T.
5-D Pond Park Road, Hingham, MA 02043
Mass: 800-442-8070 or 617-740-2510 9/87

800-421-8006

THE PROGRAMMER'S SHOP
Your complete source for software, services and answers

Object-Oriented Programming



If you've been following this column, you will have noticed a strong emphasis on object-oriented tools. That's because I've been deeply involved in the evaluation of object-oriented technology for the past year or two. I'm not alone: in the course of this evaluation, I've met many professional programmers from both large and small companies who are evaluating or actively using object-oriented systems. Large companies such as Hewlett-Packard and General Motors are using object-oriented approaches in major development work. There is a major thrust going on in the industry just now to introduce this technology into important and even critical projects.

Nevertheless, some programmers and managers are reluctant to adopt object-oriented tools. I thought it would be a good idea to devote some column space this month to some of the reasons why people have, after investigating the paradigm, abandoned or avoided it. Also, if any of you have had the experience of deciding whether to use the object-oriented approach in a major programming project, please write and tell me about your experience. I'm interested both in those in which you chose the object-oriented approach and those in which, after some serious study, you decided against it.

I also discuss this month what object-oriented programming has to offer to artificial intelligence work,

by Ernest R. Tello

and I consider what tools are needed to really exploit the object-oriented paradigm in AI.

Adventures in Object-Oriented Programming

One programmer I spoke with is Al Globus, who works at NASA Ames for the contractor Sterling Software. Al used an object-oriented C tool in a project involving simulating life patterns that might occur on the space station. Recently, Al went back to using straight C. His reasons for switching back intrigued me. But what intrigued me even more is that even though he was using regular C, he was still, in a way, doing object-oriented programming. He had abandoned the tool but not the paradigm.

The point is important and is often overlooked: it is not always necessary to use an object-oriented programming language to program in the object-oriented paradigm. Al admitted that it was primarily because he had already worked on the program in an object-oriented language that he was able to do object-oriented programming in regular C. He had got to know the code and the problem addressed so well after writing the program once that it was not all that difficult to write it again without the aid of an object-oriented programming tool.

I also spoke with Serge Hering, a software engineering consultant who evaluated an object-oriented C package as a candidate for a real-time system in an IBM PC/AT environment but decided against it. Although sympathetic toward the concept of object-oriented systems, Serge doubted that current tools and the industry at large had matured enough for the concepts to be put to use in the project he was undertaking. As he put it, "Object-oriented concepts will have to encompass all aspects of computing systems to be really practical."

According to Serge, a true object-oriented C system needs to be more than just a preprocessor to C. It requires that the object-oriented paradigm be supported at the operating system level (he's concerned with doing multitasking). After calling one of the vendors and questioning the staff on this point, he was told that object-oriented C on an AT was not a suitable vehicle for implementing a multitasking application.

His basic position now is that object-oriented programming is an all-or-nothing affair. In order for it to really work, it will have to be adopted across the industry, or attempts at using it are liable to backfire. The additional overhead required by object-oriented systems, he feels, can be handled in the proper environment, but the danger is that aspects of a system that have not been implemented with the object-oriented approach in view could come back to haunt it unless the paradigm gets the kind of industrywide support he describes.

Personally, I find Serge's view, for all its merits, a bit too conservative. The most interesting point I think he raises is just what an object-oriented C that encompassed more than just a preprocessor to C might look like. I would be interested in hearing from readers who may have an idea on what such a "total" object-oriented system in the C environment might comprise.

Object-Oriented Programming for AI

But I haven't really explained why I am concentrating so much on object-oriented programming in a column ostensibly devoted to artificial intelligence techniques. It's simple, really.

I believe that organizing programs along object-oriented lines is not only one of the best means we have

What experts are saying about PC Scheme from Texas Instruments:

". . . At less than $100, this is easily the best buy in Lisp systems for PCs."

Dr. Dobb's Journal, February 1987

Discover how powerful—and inexpensive—PC symbolic programming can be with PC Scheme from Texas Instruments. Whether you're an experienced Lisp programmer or just beginning, PC Scheme is the complete, $95* solution to your software development needs. PC Scheme combines elegant simplicity with remarkable speed in a full Lisp development system. Named *PC Tech Journal*'s Product of the Month (August 1986), PC Scheme brings professional Lisp programming features to personal computers.

PC Scheme 3.0

—Optimizing incremental byte-code compiler for ease of programming and operation
—EMACS-like editor
—Lexical scoping of variables
—Ability to suspend PC Scheme, execute DOS-based programs, then return to PC Scheme
—Random-file access and binary-file support
—Extensions for debugging, graphics and windowing
—External language interface to C, Turbo Pascal® and other languages
—SCOOPS (Scheme Object-Oriented Programming System)
—Two-megabyte extended/expanded memory support
—New manuals with tutorials and examples

Find out for yourself why experts are praising PC Scheme. For the dealer nearest you, or to order by phone, call toll-free:

1-800-527-3500

** TI Suggested list price*

PC Scheme runs on IBM® Personal Computers and compatibles (including the Texas Instruments Business-Pro™ computer). Minimum configuration: 512K RAM, dual floppy system.

Turbo Pascal is a registered trademark of Borland International. IBM is a registered trademark of International Business Machines Corporation. Business-Pro is a trademark of Texas Instruments Incorporated.

TEXAS INSTRUMENTS

261933A

(continued from page 130)

so far for achieving greater programming efficiency and modularity but also that it has important implications for the goals of AI. These potential advantages and implications do not come automatically with the use of the paradigm, but through it many of them seem to be much more attainable. Possibly one of the most important issues is the potential of object-oriented systems for alleviating the problem of "sequential closure" in programming.

What I am referring to by the term *sequential closure* is the fact that, in the current stage of software technology, if part of a program breaks, then very often the whole program breaks. This stands in contrast to most living organisms, which show considerable resiliency or "fault tolerance" to breakdowns in their functioning parts. This phenomenon seems nowhere more prevalent than in the brain. Many studies have shown the astounding degree to which the brain can adjust to massive tissue loss, for example. But, as we know, removing even a line from most programs can be fatal.

As sequential closure is so much a part of current programming technology, it might be useful to spell out why I think it is not the most desirable constraint to have. As I see it, there are three main consequences of the sequential closure of current software technology:

1. A considerable amount of programming time must be spent not only in debugging but also in redebugging.
2. There is an inherent limit to how "user-friendly" programs can be.
3. There is an inherent limitation on the flexibility and generality of problems that programs can address.

Most programmers would agree, I think, that advances are needed to improve the state of our technology in at least one, if not all three of these areas.

Object-oriented programming is not a magic fix for these problems, but there is at least one area in which, I believe, it can help significantly. The main feature of the object-oriented approach that makes this possible is the ability to accomplish a new and important degree of partitioning of large-scale functionality.

Now there is nothing new about partitioning the functionality of the program. Libraries provide this. We currently take for granted the way that reusable library functions can be written so as to be used time and time again in a variety of situations. Of course, this reusability often comes only at the expense of a great deal of effort, but it is taken for granted that it is one of the goals of competent programming.

The problem is that functions are still at a fine-grained level in the organization of most reasonably sized programs. The main sequence of code that calls the library functions is still very much subject to sequential closure. The code consists of a sequence of statements and subroutines that provides a closure on the correct functioning of the program that often must be debugged and redebugged each time important modifications are made to the program. As programs get larger and more complex (as in the case of AI), it becomes imperative that much larger functioning parts

PVCS *The Most Powerful & Flexible Source Code Revision & Version Control System.*

The POLYTRON Version Control System (PVCS) allows programmers, project managers, librarians and system administrators to effectively control the proliferation of revisions and versions of source code in software systems and products. PVCS is a superb tool for programmers and programming teams. If you allow simultaneous changes to a module, PVCS can merge the changes into a single new revision. If changes conflict, the user is notified. Powerful capabilities include: Storage and retrieval of multiple revisions of text; Maintenance of a complete history of revisions to act as an "audit trail" to monitor the evolution of a software system; Maintenance of separate lines of development or "branching"; Levels of security to assure system integrity; An intelligent difference detector to minimize the amount of disk space required to store a new version. Requires DOS 2.0 or higher. Compatible with the IBM PC, XT, AT and other MS-DOS PCs. **Maintains source code written in ANY language.**

Only PVCS meets the needs of Independent Programmers and Corporations. Once you standardize on PVCS, the "Logfiles" used to track and monitor changes are interchangable between any PVCS product. You will receive full credit for your initial purchase if you upgrade to a higher-priced PVCS.

Personal PVCS — Offers most of the power and flexibility of the Corporate PVCS, but excludes the features necessary for multiple-programmer projects. **$149**

Corporate PVCS — Offers additional features to maintain source code of very large and complex projects that may involve multiple programmers. Includes "Branching" to effectively maintain code when programs evolve on multiple paths (e.g., new versions for different systems, or a new program based on an existing program). Single User License Price. **$395**

Network PVCS — Extends Corporate PVCS for use on Networks. File locking and security levels can be tailored for each project. 5-Station License $1,000. Call (503) 645-1150 for pricing on Licenses for more than 5 Stations. **$1000**

TO ORDER: VISA/MC **1-800-547-4000.** Dept. No. 355.
Oregon and outside US call (503) 684-3000. Send Checks, P.O.s to: POLYTRON Corporation, 1815 NW 169th Place, Suite 2110, Dept. 355 Beaverton, OR 97006.

POLYTRON
High Quality Software Since 1982 ®

CIRCLE 283 ON READER SERVICE CARD

of programs have this packaged functionality so that they do not have to be repeatedly adjusted and debugged to accommodate changes and additions in other parts of the code.

Object-oriented programming can provide broader-grained partitioning. This is, of course, one of the reasons that suggested to Doug Cox, the author of *Object-Oriented Programming: An Evolutionary Approach* (Addison-Wesley, 1986), that object-oriented systems could be described as software ICs.

In the case of hardware ICs, we have units of packaged functionality that can be used in a wide variety of cases without modification or further debugging. The scope of ordinary library functions is so narrow that no one has suggested that they be compared with ICs. It is only in the most rudimentary programs that functions operate as main operating parts within the program.

Functions are usually at the same time too general and too specific to have this role. They are too general in that, rather than being a functioning part of an actual program, they are more like standard nuts and bolts that come in an enormous variety of sizes. Without any arguments supplied, with many functions it is still often unclear just what they will do.

On the other hand, functions are too specific in that the variety of situations to which they can respond is often very limited. A function usually requires that the programmer know in advance just what situation is being covered each time it is called. Functions are also specific in that the types of things they are responsible for are usually fairly detailed, low-level operations. With objects, which are usually made up of several closely related and interacting functions, we have a way around this limitation.

Consider classes. Classes in the object-oriented paradigm are a kind of IC template for stamping out custom working parts for a wide variety of programs. There may be only a small number of instances of that class that are actually used in a given program. But in making those instances, it is typical to assign values to instance variables that define the conditions of a problem once and for all while still leaving open the issue of just what messages will be sent to what objects at what time. What can be set without hand coding by the programmer, however, is what arguments these methods will take in a lot of cases. It is possible to have a systematic way of assigning whole sets of arguments to methods simply by setting values to instance variables at the time an object that acts as a functioning part of a program is initialized. I hope in a later column to provide some actual coded examples of this.

Inference Engines for Objects

I believe that the object-oriented approach has a lot to contribute to furthering the goals of AI. Using object-oriented systems to model problem domains can be a little deceptive, though, unless some crucial design issues are met. Simply creating a static model of the unique aspects of a certain "world" of activity, such as the world of medical diagnosis or electronic troubleshooting, is no guarantee that programs using such models will be at all intelligent.

In other words, if the information

THE 150% SOLUTION FOR SUPERIOR DATABASE DEVELOPMENT AT 62% OFF.

PHACT-manager™ gives MS-DOS™ programmers the ISAM they need. Plus a Report Writer, Query Language and full C source code.

- Efficient B + Tree access method.
- Unlimited number of keys and variable length records.
- Security: Password protection, shared/exclusive use.
- Runs on networks.
- Sequel-like query language for interactive or batch query/update.
- Report Writer: Perform "joins," create and use variables, sort, format and more.
- Versions for all popular C compilers.
- Thousands of licenses sold.

To order or get more information, call us at
1-800-222-0550 (outside NJ) or 1-201-985-8000 now.
MasterCard/Visa accepted.

Only $249 complete! Full C source code.
No royalties!

UniPressSoftware

UniPress Software, Inc. 2025 Lincoln Highway Edison, NJ 08817

MS-DOS is a trademark of Microsoft. PHACT-manager is a trademark of PHACT Associates.

CIRCLE 77 ON READER SERVICE CARD

ARTIFICIAL INTELLIGENCE
(continued from page 133)

just sits there statically, it does not contribute the kind of knowledge we are interested in, no matter how well the world model may have been designed. Everything depends on the techniques used for making use of that information. The more actively a program can be designed so as to be continually accessing the knowledge incorporated in a hierarchical model of a problem space, the more intelligence will have been gained.

In rule-based systems, the inference engine is a general program that accesses knowledge in the form of rules and keeps things moving so that the knowledge captured in rules gets used and applied. But so far, there have not been any general-purpose "inference engines" built for accessing knowledge in the form of objects. As things now stand, we are just beginning to get the idea of how to use these structures in a very ad hoc way. It is worthwhile, though, to speculate a little bit on what generic routines might be important for AI programs using objects to create deep models of domains.

Certainly some unusual, yet generic, search routines specific to object-oriented systems are a necessity. One such basic routine would be a method that built its own list of all the instances of a given class for the purpose of search, then made its own ordering of the names of these objects, and then searched them for specific information by looking up the values of various slots or instance variables. Far more intricate and flexible search routines are possible, too, such as those that can do a more general type of information gathering so as to provide a temporary seach space of only the relevant data needed for a given situation. This idea of programs that can build temporary search spaces dynamically for use in solving problems efficiently is one that has a broad applicability to several different types of AI application.

An interesting use of class hierarchies in AI has been the partitioning of rule-based knowledge into sets of rules that are organized according to a network of situations in a problem space. Here only the tip of the iceberg has been touched so far. There is room for a wide range of approaches to providing flexible problem solving methods based on sophisticated routines for selecting rulesets to apply by rapid initial searches up and down a hierarchy rather than just by executing simple test and branch procedures.

For example, let's say that the application is one of actually building a design for something. Here, one of the important services that computer technology can provide is in helping human designers make sure they have not overlooked anything important. The problem is that what is important in a given design can differ enormously from case to case, depending on the details and requirements of a given design. In such a case it would be useful to have various sets of constraint rules that could be selected and applied in a selective way rather than by a rigid procedure.

One of the key types of know-how that human designers learn by experience is how much detail is relevant for evaluating a given aspect of a design. This is important for avoiding entirely inappropriate analyses and conclusions. So, for example, in some designs, there is enough standardization in the parts being used that an iteration through each part in turn would be absurd. All that is needed is inspection of a few typical elements to reach an important general conclusion. This is the kind of realization that a human designer reaches by common sense but which can be automated. In this case, a general routine could be used that sifted quickly through all the instances of each type of design element used in a proposed design and decided which categories would need a detailed, iterative treatment and which could be evaluated by inspecting typical elements.

The general point that object-oriented systems can be effective for AI applications only to the degree that an active access is provided to the information represented as objects has some interesting consequences. For one thing, it means that an adequate object-oriented programming environment must provide at least some minimum features to make this possible.

First on the list is the ability to keep track of all the objects currently in the system, which includes both classes and instances, according to their place in the hierarchy. That's more powerful than it sounds; part of what we consider simple common sense, which all normally functioning humans have, is a sense of the place of things in an implicit logical or conceptual hierarchy. It's also what conventional programs don't have.

For example, we take it for granted that bicycles and cars are a means of transportation and hence are in the category of equipment or artifacts and that, therefore, certain kinds of behavior with them are appropriate. The boundaries of this are subject to challenge, of course, but we take it for granted that it is not appropriate to use a piano for firewood or to risk our lives to rescue a bookend from a burning house. We know that liquids cannot be safely wrapped in newspaper and millions of other bits of general, practical knowledge that seem to be stored as a general understanding of types of things and their roles and characteristics.

The logical way to represent this common sense knowledge of how to get around in the practical world is as some kind of conceptual hierarchy. That way, the mere fact that a given object is an instance or member of a class will activate certain knowledge about how to deal with objects of that type.

The two main ways this is currrently done in AI are by use of rule-based programming and procedural attachments or active values. In many of the larger professional AI programming systems, it is possible to store rules about classes of things that are invoked for certain types of objects. The problem, though, is to provide a means of determining when it is appropriate for such knowledge to be invoked. Active values or procedural attachments act like "demons" that are waiting for certain events in order to be invoked. So various operations can be waiting in the wings, as it were, for certain values to be changed or accessed.

Software-ICs®

Building all new software from scratch is not the most productive way to work.

All of us who have built a major software system have felt that we have built the same piece of code before, especially when we have been involved in successive projects.

Many of us have had a strong interest in 'reusable' software. We have tried cataloging components, developing function and subroutine libraries, etc. Our expectations have not been met.

Why not? We needed a small but important change—object-oriented technology.

Software Evolution

PPI® believes that the most effective way to construct reusable software, and of higher quality, is to build software components using object-oriented programming; software components designed to allow change without the adverse ripple effect typical of traditional subroutine techniques.

PPI describes these software components as Software-ICs.®

Today, few people would think of building a computer from scratch. In the future, few people will think of building software from scratch. PPI believes that the software industry is evolving, and the vehicle will be Software-ICs.

Integration

As you adopt object-oriented technology, PPI supports you with the tools and training to achieve a fully productive development environment.

PPI's hybrid-language approach lets you build on your current software investment. You don't have to learn a new language to experience the benefits of object-oriented technology.

Real Solutions

PPI is the leader in providing object-oriented development tools and ready-to-use ICpaks.™

For example, ICpak 201 is a set of 49 pretested Software-ICs which enables the developer to build sophisticated, multiwindow user interfaces for workstation applications.

Today, PPI products are in use worldwide, by multi-billion-dollar international companies, universities and government agencies. Projects involving more than 50 person-years of effort and over 100,000 source lines of code have been successfully completed, ahead of schedule, and are in commercial use.

To learn how PPI products, training and service can help you meet your development goals faster, with higher quality results, call our European representative in Stockholm at 8-62-65-08, or call us in the United States at (203) 426-1875, Telex: 506127.

Or write PPI, 75 Glen Road, Sandy Hook, CT 06482.

PPI®

Software to Build Software

CIRCLE 390 ON READER SERVICE CARD

(continued from page 134)

So, for example, if we attempt to change the physical state of an artifact, such as by burning a piano, before that operation can be permitted, certain procedures can be automatically invoked that apply knowledge about appropriate behavior for that class of things. A demon for wooden objects might state that things in the category of useful furniture have a greater value than satisfying temporary needs for warmth by destroying them.

When representing knowledge to be used to solve certain problems, you are constantly faced with decisions about how implicit or explicit to make the knowledge. In conventional programming the knowledge that is incorporated in a program is nearly always completely implicit. That is, it does not really exist as knowledge in the program other than as knowledge used by the programmer to design its functioning.

In AI paradigms such as rule-based programming, the attempt is made to make knowledge explicit as general rules so that this knowledge can be applied to new situations not entirely foreseen by the developer. The ability to handle novel situations is still limited, however. So far, the main advantage gained by making knowledge explicit is that greater generality and flexibility can be given to applications. Rather than having the knowledge "hard-wired" into a conventional program that is not capable of comparing different pieces of knowledge to reach its own conclusions, it is useful to provide a way of modeling the general process by which practical results are obtained. The alternative is just using the results in a purely ad hoc way.

Obviously, it is never possible to make everything the result of a process of explicit reasoning. Even humans could not function that way. Nothing would ever get done in real time. We would all end up like parodies of Hamlet, forever analyzing the pros and cons of every aspect of even the most trivial actions. One of the rules of thumb for practical activity, whether by human or machine, is that certain things must be taken for granted as implicit and used for a limited number of events of explicit analysis or reasoning.

From this point of view, we can see one of the main distinguishing features of human as opposed to machine analysis. At any given time, we can decide to change something from the status of implict to explicit if we need to for any reason. It's an important part of living to sense when it is necessary to question things normally taken for granted. So far, with AI systems, the decision about what knowledge will be implicit and what explicit is decided by the programmer and cannot be changed except by rewriting the program. In a future column I will describe a program I have been designing that actually decides to some degree.

So, as things now stand, the main trade-off, as always, is flexibility vs. efficiency. The question is how much knowledge can be made explicit without losing adequate performance. This is always problem specific. For some problems, a week of processing time would be acceptable if the result obtained were accurate and reliable enough. For others, even ten seconds might seem an eternity. Making the proper choice between what to make explicit and implicit in an AI program is often something that can only be done by developers with a great deal of experience in the field. Even here, often decisions made early on turn out to be proven wrong later in the game.

The advantage of modular paradigms such as rule-based and object-oriented systems, though, is that changes can usually made by local modifications that do not alter the functioning of the program as a whole. Often knowledge can be made explicit by adding and subtracting various rules and/or functioning classes that do not require that the application be substantially rewritten. And that sort of functional partitioning is essential to any large-scale programming project and especially to work in artificial intelligence.

DDJ

Vote for your favorite feature/article.
Circle Reader Service **No. 9.**

The C Programmer's Assistant

C TOOLSET™

Unix-like Utilities for Managing C Source Code

No C programmer should be without an assistant. C ToolSet provides you with the support you need to make C programming easier.

All of the utilities are tailored to the C language but you can modify them to work with other languages as well.

Source code in standard K&R C is included. You are welcome to use it with any compiler (UNIX compatible) and operating system you choose.

The documentation contains descriptions of each program, a listing of program options, and a sample run. On-line help responds to -? on the command line with a list of options

12 Time Savers

DIFF - Compare files line by line; use CMP to compare byte by byte.
GREP - Regular expression search.
FCHART - Trace the flow of control between large program modules.
PP - Format C program files so they are easier to read.
CUTIL - General purpose file filter.
CCREF - Cross reference variables.
CBC (curly brace checker) - Check for pairing of curly braces, parens, quotes, and comments.
Other utilities include *DOCMAKE, ASCII, NOCOM,* AND *PRINT.*
Requires MSDOS and 12K RAM

MONEYBACK GUARANTEE

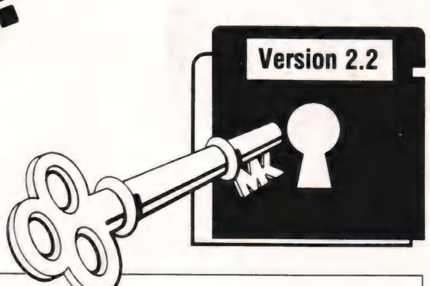
Try C ToolSet ($95) for 30 days — if not satisfied get a full refund.

Call (800) 255-4659
In MA (617) 331-0800

The Coder's Source™ 541-D Main St., Suite 412, So. Weymouth, MA 02190

CIRCLE 152 ON READER SERVICE CARD

MASTER*KEY Unlocks Everything!



turn this

```
C:>DEBUG PROGRAM.COM
-D100 136
8848:0100  EB 18 49 6E 63 6F 72 72-65 63 74 20 44 4F 53 20   k.Incorrect DOS
8848:0110  76 65 72 73 69 6F 6E 0D-0A 24 50 B4 30 CD 21 86   version..*P40M!.
8848:0120  E0 3D 36 01 72 05 3D 0A-02 76 09 BA 02 01 B4 09   `=6.r.=..v.:..4.
8848:0130  CD 21 CD 20 58 EB 2F                              M!M Xk/
-Q
```

into this!

```
H00100: JMP     Short H0011A                            ;00100 EB18              --
;-----------------------------------------------------
        DB      "Incorrect DOS version"                 ;00102 496E636F727265
        DB      0Dh                                     ;00117
        DB      0Ah                                     ;00118
        DB      "*"                                     ;00119 24
;-----------------------------------------------------
H0011A: PUSH    AX                                      ;0011A 50                P
        MOV     AH,30h                                  ;0011B B430              _0
        INT     21h             ;1-DOS_Ver_Number       ;0011D CD21              _!
        XCHG    AH,AL                                   ;0011F 86E0              __
        CMP     AX,0136h                                ;00121 3D3601            =6_
        JB      H0012B          ; . . . . . . . . . .   ;00124 7205              r_
        CMP     AX,020Ah                                ;00126 3D0A02            =__
        JBE     H00134          ; . . . . . . . . . .   ;00129 7609              v_
H0012B: MOV     DX,0102h                                ;0012B BA0201            ___
        MOV     AH,09h                                  ;0012E B409              __
        INT     21h             ;1-Display_String       ;00130 CD21              _!
        INT     20h             ;TERM_normally:20h      ;00132 CD20              _
;-----------------------------------------------------
H00134: POP     AX                                      ;00134 58                X
        JMP     Short H00166                            ;00135 EB2F              _/
;-----------------------------------------------------


MASTER*KEY XREF - PROGRAM.XRF                                            Page 1

0102h                  :    121   2F5   301   320
020Ah                  :    126
03CBh                  :    12B
1-Display_String       :    130   591   610
1-DOS_Ver_Number       :    11D
H00100                 :    100
H0011A                 :    100   11A
H0012B                 :    124   12B
H00134                 :    129   134
H00166                 :    135
TERM_normally:20h      :    132
```

NOTE: The cross-reference is by memory location within the program file!

NOTE: The output is totally Microsoft MASM-compatible.

(not copy protected)

MASTER*KEY
No Other Product Comes Close!

An EXPERT may not know the solution, but always knows where to find it.

MASTER*KEY HELPS ANYONE solve those confusing and frustrating software puzzles more rapidly and easily than any other software available, at any cost! It gives you know-how within hours that may otherwise take years of experience. Create a new program from an old one. DON'T REINVENT THE WHEEL!

MASTER*KEY - Smart!
MASTER*KEY is an intelligent self-documenting MS-DOS reverse assembler. Its sophisticated procedures swiftly race through massive and baffling object code files to effortlessly discover potential trouble spots.

MASTER*KEY - Educational!
YOU DON'T NEED TO KNOW ASSEMBLY LANGUAGE! MASTER*KEY will take any program from your IBM-compatible computer and return fully-documented, easily-understood assembly language source code (Microsoft MASM 4.0 compatible).

MASTER*KEY - Easy To Use!
MASTER*KEY works both automatically from the DOS command line or interactively from menus similar to Lotus Corporation's 1-2-3 or Symphony. No need to remember any new commands or continually refer to a manual. Use it immediately!

Minimum System Requirements:
256K + 8088/8086/80186/80286/80386 PC
MS-DOS or PC-DOS 2.0 +
One 360K DSDD Floppy Drive (IBM PC Format)

MS-DOS is a trademark of Microsoft.
PC-DOS is a trademark of IBM.

MASTER*KEY will guide you step by step to:

1. Help you learn assembly language, if you desire.
2. Discover how any program runs or why it doesn't.
3. Alter or remove unwanted object code from any program.
4. Incorporate routines from compiled programs into other assembly language, Basic, C, or Pascal programs.
5. Make software more compatible with your computer. Be certain a questionable program won't damage your system BEFORE you run it.
6. Modify software to operate with other versions of DOS.
7. Customize your COMMAND.COM or other executable program directly or by reassembling your altered MASTER*KEY source code.

Order Now! Just $79⁹⁵

Phone orders accepted on MC or VISA
Send checks to:
Sharpe Systems Corporation
2320 E Street, Dept 44, Laverne, CA 91750

$82.45 (includes $2.50 shipping)
$87.65 in California (includes tax & shipping)
C.O.D. orders add $2.00

(714) 596-0070

Dealer/Distributor Inquiries Welcome

Sharpe Systems Corporation
2320 E Street, Dept 44, Laverne, CA 91750

CIRCLE 176 ON READER SERVICE CARD

MASTER*KEY should not be confused with any public domain or share ware software that may have a similar name or be a similar product.

(continued from page 12)

type unevenly, the average typing speed was often much less than ten characters per second.

In spite of their slowness the teletype machines were the most practical terminals for multiuser systems (then called time sharing systems) and for small computers. The electronic terminals using a CRT were at that time dumb terminals and very expensive. Furthermore, if hard copy was desired it was necessary to buy a printer which would cost almost as much as a full teletype terminal. The widespread use of teletype machines as terminals left its mark on computer standards in many ways besides the TTY abbreviation. The most important is probably the ASCII code. This was developed for the teletype machines and then adopted by computers when the teletype machines were adopted as terminals.

Many computerists are puzzled by the fact that all of the control codes except DEL are at the beginning of the codes while DEL is at the very end. The old name for DEL, RUBOUT, gives a clue to this mystery. Because it was very difficult to type at anywhere near the maximum of ten characters per second on the teletype machines most teletypes were equipped with a paper tape punch and reader. The message was punched into paper tape off line and then transmitted at the maximum rate by the tape reader on line to save expensive on line time. But suppose you made a mistake and hit the wrong key near the end of punching a long tape. Once a key was struck there was no way to unpunch the holes in the tape and it would be very frustrating to have to do the whole tape over again. Instead the tape was manually backed up to the error and the error overpunched with the RUBOUT code which punched all the hole positions. The receiving teletype was set to ignore RUBOUT codes so that the errors simply didn't print.

The paper tape units came in at least two version, the 7-bit and the 8-bit types. The 7-bit type was common for most communications uses but for computers and for communication channels with a parity bit the 8-bit units were used. With computers paper tapes were used for off line storage and for distributing software. For computer use at the computer high speed paper tape punches and readers were developed and became so common that CP/M had input/output channels reserved for them.

Many of the earlier and less expensive teletype machines were upper case only and this is one of the reasons that BASIC as developed at Dartmouth College would work with all upper case. Apparently the Bell Labs could afford the more expensive teletypes with the full character set and both C and Unix are case sensitive although Unix has a provision for translating upper case to lower case for upper case only terminals and both tend to use mostly lower case. Some teletypes could transmit either case although they printed upper case only, translating lower case to upper case when receiving.

Note that all of the older text editors such as ed on the Unix system are line editors and that only the newer editors such as vi are full screen editors. That is because with the teletypes as terminals there was no screen.

This is not the first case where decisions made in the past for valid reasons persist in standards long after the reasons for making those decisions no longer apply. After all the gauge of modern railroads was set by decree of an ancient Roman emporor and the layout of the standard typewriter keyboard was made to discourage fast typing and thus avoid key pileups. After a standard has been in use for a while there are many things around embodying that standard that would be obsolete if the standard were changed so old standards linger on long after they are obsolete.

Incidentally the slow teletype machines hooked to a timesharing system seemed like a big advance to programmers who previously had to punch their programs on a card punch and submit them for batch

The
**Heap Expander™
version 2.0**

Now your programs can have virtually unlimited heap space using expanded memory, extended memory, disk space, or any combination of the three. And it's all transparent. The Heap Expander's startup code checks the system's resources and uses whatever is available.

still
$59.95*

- Uses LIM-standard expanded memory if present.
- Uses AT-style extended memory if present.
- Swaps data to disk as needed.

Libraries and Source Code for:
- Turbo C
- Microsoft C 4.0 and 5.0
- Mark Williams C 3.1 and 4.0
- Lattice C 3.2
- Turbo Pascal 3.0 and 4.0
- Logitech Modula-2/86

Requires an IBM PC, XT, AT, or close compatible with MS-DOS or PC-DOS version 2.0 or above

MC/VISA/COD call
1-800-248-1045 x 100 (US)
1-800-952-5560 x 100 (Idaho)

*Idaho residents add 5% sales tax
Foreign customers add $4.00 for shipping and handling.

The Tool Makers
P.O. Box 8976
Moscow, Idaho 83843
208-883-4979

CIRCLE 319 ON READER SERVICE CARD

the time you spend
creating code
is simply
time
slipping
away.

Which is
why there's
Screen Maker™
from SoftScience.
Only Screen Maker generates
Turbo C, C, Turbo Pascal, dBASE,
Standard BASIC, Quick BASIC, and Turbo
BASIC source codes. Each code is incorporated into your program, reproducing your
screen exactly as you've drawn it. PC Week called
it "...the perfect programming utility. It does its job
and saves the user time." And at $79.99, you will save
more than time. Call SoftScience toll free at 1-800-453-5000.

SoftScience Corporation, P.O. Box 42905, Tucson, Arizona 85733-2905.

A Different BASIC Might Make All the Difference

We'll skip the four-color gatefold. And the extravagant claims.
Because if you're serious about programming, you just want the straight facts:

	True BASIC 2.01	Microsoft Quick Basic 3.0	Borland Turbo Basic 2.0
GRAPHICS			
Supports Hercules Graphics	YES	NO	NO
Device-Independent Graphics Syntax	YES	NO	NO
User-Defined Coordinates	YES	LIMITED	LIMITED
Matrix Graphics Coordinates	YES	NO	NO
ARRAY HANDLING			
Matrix Algebra	YES	NO	NO
Maximum Numeric Array	UNLIMITED	64K	64K
Max. Number of Array Dimensions	255	63	8
Max. Number of Elements/Dimension	UNLIMITED	32K	32K
Dynamic Redimensioning	YES	NO	NO
Matrix I/O Statements	YES	NO	NO
STRING/FILE HANDLING			
Maximum String Length	64K	32K	32K
Total String Space	UNLIMITED	64K	64K
Maximum Record Size	16MB	32K	32K
Max. Bytes/Binary File Read	64K	NA	32K
PRODUCTIVITY ENHANCERS			
Modules	YES	NO	NO
Separately Compiled Libraries	YES	LIMITED	NO
Workspaces	YES	NO	NO
Immediate Mode	YES	NO	NO
SPECIAL FEATURES			
Stop/Continue Execution	YES	NO	NO
Max. Source File	UNLIMITED	UNLIMITED	64K
Script Files	YES	NO	NO
Keystroke Macros	YES	NO	NO
Max. Characters/Line	64K	255 char.	249 char.
Max. Scalar Data Space	UNLIMITED	64K	64K
Mouse Support	YES	NO	NO
80386 Version	YES	NO	NO
Portability to:	Macintosh, Amiga, Atari	Translation required	No other machines

NOW ONLY
$99.95

Three very structured, very powerful programming packages. All with fancy editors and fast compilers. Two of them are the same in other respects. And one of them, True BASIC, is quite a bit different. With syntax and features that will make you more productive.

That's why reviewers for magazines like BYTE, PC Tech Journal and Computerworld keep giving True BASIC their top marks. And why OEMs pick True BASIC after they've evaluated all the others. See why True BASIC can make the difference for you.
Call **1-800-TRBASIC** today.

True BASIC

39 SOUTH MAIN STREET
HANOVER, N.H. 03755
(603) 643-3882

True BASIC, Quick Basic and Turbo Basic are trademarks of True BASIC, Inc, Microsoft and Borland, respectively. Macintosh, Amiga and Atari are trademarks of Apple Computer, Inc., Commodore-Amiga, Inc. and Atari Corporation. Copyright 1987 True BASIC. Specifications are accurate as of August 1987.

CIRCLE 344 ON READER SERVICE CARD

(continued from page 138)

processing every time they wanted to try something. Then they would wait hours or even days for the results which might be only a cryptic error message. Think how that would affect your debugging procedures.

David S. Tilton
27 Pennacook St.
Manchester, NH 03104

Interrupts

Dear *DDJ*,

Thomas Zimniewicz wonders (in his article "An Extended IBM PC COM Port Driver," June 1987) why the IBM PC serial ports use a UART bit to enable the interrupt-driver line. They do this so the interrupt can be turned off. Normal computers (Z-80s, etc.) often have such lines driven low by open-collector drivers when signaling an interrupt, so that more than one device can use the physical hardware interrupt wire (the software, in such a case, would be obliged to poll each possible device). Such interrupt lines are "pulled-up" somewhere in the system with a single resistor, so even if nobody is driving the interrupt wire, the CPU won't see erroneous interrupts.

The IBM/Intel hardware, on the other hand, drives the interrupt line high when an interrupt occurs, and the hardware has no inherent mechanism for sharing an interrupt line. The serial cards ameliorate this to some extent by providing a disabling/enabling mechanism attached to a spare bit on the UART; thus, by selectively turning off one guy's interrupt and turning on another, more than one card can use the same interrupt wire; but not really at the same time. The serial cards usually also include a DIP switch enable/disable, which is basically in series with the electrical mechanism.

Here are some of the nice things that can happen if you forget about the interrupt enable bit.

Two or more serial cards—or other cards?—could have their interrupts enabled simultaneously on the interrupt line. This is an electrical short; there probably won't be any smoke or anything, but the interrupts definitely won't work the way

The Software Family

649 Mission Street
San Francisco, CA 94105

(1)800-443-7176

In California or outside U.S.

(415)583-4166

Basic
Turbo Pascal
Publishing

C
dBase
AI

Trust TSF to provide the best value and service!

* Technical advice and support by programmers *

* Honest and equitable business practices *

* Prompt, flexible service to meet your needs *

Turbo Pascal

We will be shipping Turbo Pascal version 4 products as they become available. Specify your version preference when ordering.

Turbo Pascal (list ~~$99~~)...**$59**

Turbo Toolboxes from Borland can save you save hours of coding. Specify Database, Editor, Graphix or Gameworks.

Toolbox (list ~~$70~~) **$49**

Want to stay with Turbo Pascal version 3? TSF stocks a complete line of add-on products that let you get v4 capabilities using the Borland's v3 compiler. Call for information concerning products for v4.

Thirty day money back guarantee on all Borland products.

Debugger

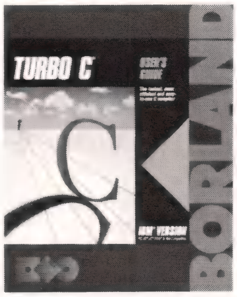
TDebug+ v2 from Turbo*Power speeds debugging by letting you peek into memory while your program runs. A simple menu interface lets you control execution flow, examine/change variables by name, and view the source code corresponding to the currently executing instruction. New features include conditional breakpoints that stop execution based on a variable value or memory address range and watch windows that display the contents of critical variables. Includes all source code. Thirty day money back guarantee.

TDebug v2 (list ~~$60~~).......**$48**

Other Pascal Products

Turbo Optimizer (~~$75~~)	**$59**
Optimizer w/src (~~$125~~)	**$99**
TMark (~~$80~~)....................	**$69**
Mach2 (~~$69~~)..................	**$55**
TurboWindow/P (~~$99~~)...	**$79**

Turbo C



Turbo C (list ~~$99~~)...........**$59**

Data Entry Windows

Vitamin C from Creative Programming provides a complete set of C functions for window management, screen display, data entry and editing. Vitamin C can be used by itself or with VC Screen which automatically generates C code from screens that you "paint" in full screen mode. All source included for Vitamin C. Thirty day money back guarantee.

Vitamin C (~~$250~~).......	**$159**
VC Screen (~~$99~~)	**$79**

Turbo C Functions

Turbo C Tools from Blaise Computing provides a complete set of C functions to help you write terminate and stay resident (TSR) programs, display text in overlapping windows, and write MSDOS critical error handlers (so your users never see "Abort, Retry or Ignore" again). Also includes support for interrupt service routines (ISR), fast video display and directory management. Includes all source code. Thirty day money back guarantee.

Turbo C Tools (~~$129~~).. **$99**

Other C Products

Blackstar C Functions (llist ~~$129~~)	**$99**
PC Lint (~~$139~~)	**$103**
Periscope II (~~$1175~~)...	**$139**
TurboWindow/C (~~$99~~) ..	**$79**

Other Products

Turbo Basic (~~$99~~)..........	**$59**
Quick Basic (~~$99~~)..........	**$65**
Quick C (~~$99~~)	**$68**

Call or write for our free 20 page newsletter and catalog.

We accept checks, Visa, MasterCard, American Express and COD. We charge your card only when we ship and do **not** add a surcharge. We provide free UPS delivery on software orders over $100 ($3 delivery charge on orders under $100). UPS and Federal Express air service available. Our COD fee is $2. We carry a large inventory and ship promptly, so you usually receive your shipment within 3 to 6 working days of placing your order.

CIRCLE 230 ON READER SERVICE CARD

(continued from page 140)

you expected. You could set-up the interrupt controller properly, but leave the interrupt disabled. According to my *Technical Reference*, this leaves the interrupt line floating, i.e., your interrupts will become phase-of-moon dependent.

Perhaps with PS-17 or whoever, IBM will come up with sophisticated hardware comparable to the average microwave oven controller of today. We can but hope.

J.G. Owen
35 Admiral St.
Port Jefferson Station, NY 11776

Bandwidth Bottleneck

Dear *DDJ*,

I have noted the thoughts on memory and bandwidth in recent issues of *DDJ*; they're forward-looking and pertinent to an issue which is of increasing concern to programmers and hardware designers alike. In particular, the editorial in the March issue got me thinking with the reference to commuting. How many times have I myself sat in my car, creeping along, wishing for "more bandwidth?" Then, as the highway crews build in more bandwidth, more cars come out of the woodwork to clog it all up again.

The central question becomes not how, but why. Why should massive amounts of data be thrown from storage to processor, often in order to locate or operate on a minuscule fraction thereof, and then thrown back again, or even dumped? As long as processing power is located over here and data over there, data volume will always grow to overwhelm the physical channel, similar to budgets and pocketbooks.

Two approaches to this problem seem to point in fruitful directions: more efficient data transfer and eliminating data transfer entirely. The latter is not necessarily the logical extension of the first.

Using current technology and its extrapolations, it seems possible to explore more advanced methods of storing and indexing data so that the patterns inherent in the data are more "visible" to the data user. If the system wishes to use data for inferences or "inference-primitives," rather than as fragments which must be assembled into inferences each time or when data will be transferred, distill it first. The extra processing time up front would greatly reduce operations time later.

Eliminating data transfer, however, would require a more fundamental change in the way data and processor work together. Perhaps we are seeing the beginnings of this quantum leap in the advancing neural-network research. Why must Mohammed go to the mountain? Make Mohammed be the mountain! Data essentially becomes the processor and the problems associated with transfer to and from processing centers disappears. Analog computers operate in this fashion; the construction of the computer is really data to be operated on. The computation is essentially instantaneous without the need to read or otherwise handle data as a separate entity.

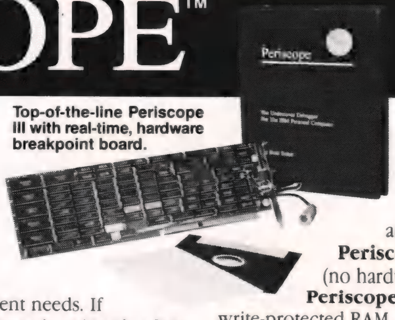
I suppose that the issue revolves around the old "chicken-and-the-egg" dilemma. If we can eliminate the need to handle all of the data, perhaps we would know the answer before asking the question, or the solutions might fail to be general enough to be truly useful. Additional research on what data really is will help us find our way.

Thanks for placing such an important issue before the informed and active computing public of *DDJ* readers. I'm sure that there will be much informed comment and discussion.

H. Ward Silver
RBR Engineering
P.O. Box 1608
Vashon, WA 98070

DDJ

PERISCOPE™ POWER



Top-of-the-line Periscope III with real-time, hardware breakpoint board.

...Keeps you going full steam ahead when other debuggers let you down. With four models to pick from, you'll find a Periscope that has just the power you need.

. . . Start with the model that fits your current needs. If you need more horsepower, upgrade for the difference in price plus $10! And don't worry about having a lot more to learn . . . Even when you move to the most powerful model, Periscope III, an extra dozen commands are all that's involved.

Periscope's software is solid, comprehensive, and flexible. It helps you debug just about any kind of program you can write . . . thoroughly and efficiently. Periscope's hardware adds the power to solve the really tough debugging problems.

Periscope requires an IBM PC, XT, AT, or close compatible (Periscope III requires hardware as well as software compatibility); DOS 2.0 or later; 64K available memory; one disk drive; an 80-column monitor.

Periscope I includes a half-length board with 56K of write-protected RAM; break-out switch; software and manual for $345.

Periscope II includes break-out switch; software and manual for $175.

Periscope II-X includes software and manual (no hardware) for $145.

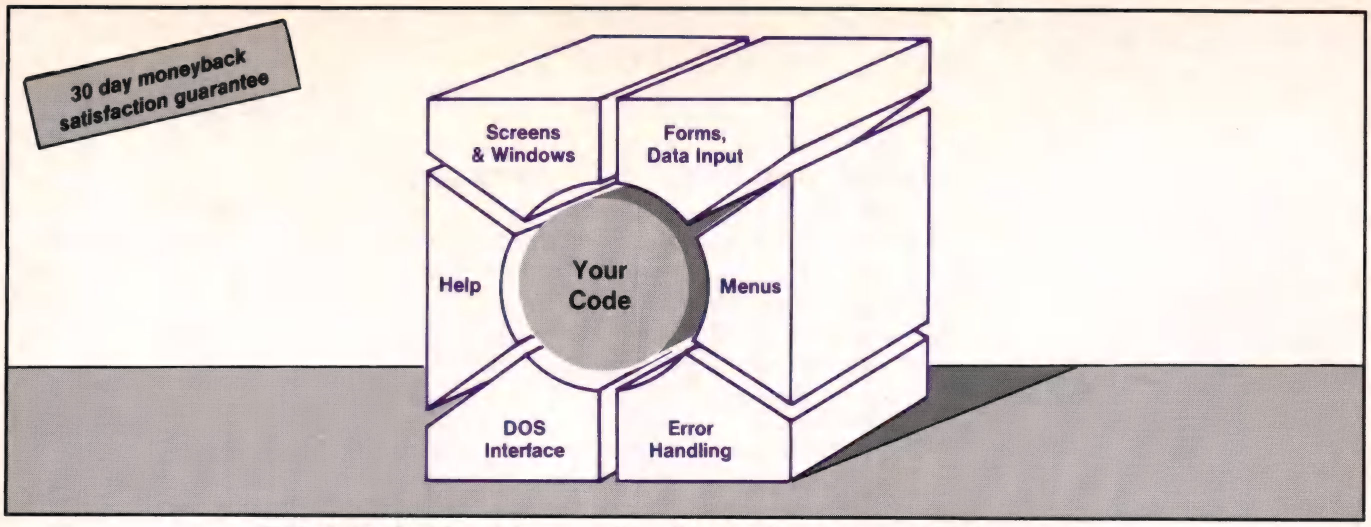
Periscope III includes a full-length board with 64K of write-protected RAM, hardware breakpoints and real-time trace buffer; break-out switch; software and manual. Periscope III for machines running up to 8 MHz is $995; for machines running up to 10 MHz, $1095.

Call Toll-Free for free information or to order your Periscope today!

MAJOR CREDIT CARDS ACCEPTED. **800-722-7006**

The PERISCOPE Company, Inc.
14 BONNIE LANE
ATLANTA, GA 30328
404/256-3860

CIRCLE 214 ON READER SERVICE CARD



C-Worthy Interface Library helps you smoothly pull together all aspects of an excellent Human Interface.

C Programmers: Wrap an Exciting, Bullet-Proof Interface Around Your Code Quickly.

Introducing... C-Worthy® Interface Library

The only human interface package you need. That's what our customers are telling us. One early adopter, Novell, Inc. uses it exclusively in the development of their NetWare® Utilities, which reach over 500,000 users. You see, C-Worthy Interface Library is the only library available to handle every aspect of your program's human interface, all in one package. Now your programs will have a consistent look and feel. You no longer have to integrate pieces of libraries from different manufacturers.

As important as you know users are, you often don't have the time to heavily invest in writing routine code. And that's OK, because with over 400 tight, ready-to-use functions, C-Worthy Interface Library takes care of the tedium and lets you spend your time doing what you enjoy. Concentrate on the heart of your application — features that make it unique, special. Let C-Worthy Interface Library do your:

- Menus
- Error Handling
- DOS Interface
- Context Sensitive Help
- Screens, Windows
- Forms, Data Input (optional)

You control color, size, border, location, etc. And if there's anything you want to change, you can. Source is available to provide you with the flexibility you need. And you can distribute your applications freely, with no royalties.

C-Worthy Interface Library requires hard disk media with 256K RAM. MSDOS 2.0 + and IBM PC, or compatible, TI Professional, NEC APC III, or VICTOR 9000. C-Worthy is a registered trademark of Custom Design Systems, Inc.

Tech Specs

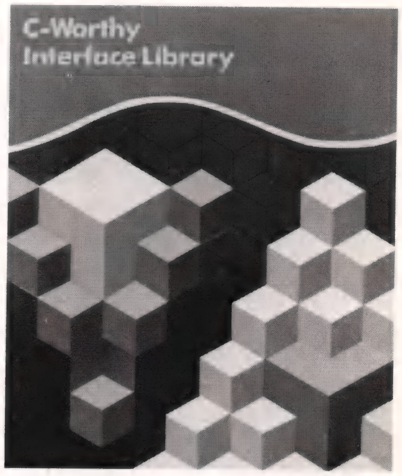
- Compilers: Microsoft 3.0+, Quick, Turbo, Lattice. All models.
- 350+ functions written in C, 75+ in Assembler.
- Menus: Fully support pop-up, Lotus style, MS Windows style (pull-down), pull-up.
- Errors: DOS, program, and user.
- DOS Interface: 62 functions. File handling, dir. and drive management, date & time conversion, wildcards, more.
- Help: System and context sensitive.
- Screens: Screen display, color palettes, save, restore, scroll, more.
- Windows: Exploding, tiled, pop-up, overlapping. Direct video access and virtual. Up to 50 active at any time.
- Keyboard Handling: Regular, function, interrupt, background procedures.
- Editing: String and word wrap text.
- Form Interface Library: 118 functions. Over 15 field types, and user definable field types. 3 levels of data validation: type, multiple field ranges, optional validation procedures. Hide, lock, or secure a field. Optimal field movement.
- Foreign Languages: All text messages in separate files for easy translation.
- Compatible with MS Windows.
- OS/2 special overlay when released.
- Machines: Autodetect for MDA, CGA, EGA, VGA, TI, AT&T, Victor.
- No royalties.

"I heartily recommend this package."
— David A. Schmitt, president, Lattice, Inc.
Over 400 developers in 16 countries already use it.

Thorough Documentation

Indexed alphabetically and by category, the 700+ page Reference Guide includes for each function: an example, description, calling conventions, return values, and related functions. The 250 page User's Guide gets you going with its tutorial and "Getting Started" sections.

"C-Worthy is a comprehensive C library whose time has come. I heartily recommend it as your next purchase." —Computer Language, 8/87



C-Worthy Interface Library:

Object only	$ 195
Form Interface Library add-on	$ 100
Object with Forms	$ 295
Object with Forms & Library Source	$ 495

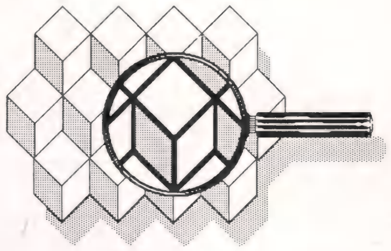
Please specify compiler and version when ordering.

To Order Call
(800) 821- 2492
in MA (617) 337-6963

Solution Systems™
541-D Main Street, Suite 410
South Weymouth, MA 02190

CIRCLE 296 ON READER SERVICE CARD

OF INTEREST



Sigma Designs has announced SigmaVGA, a high-resolution graphics board. As its name implies, SigmaVGA offers compatibility with IBM's new Video Graphics Array (VGA) standard. It also offers compatibility with the IBM monochrome display mode and CGA, EGA, and the Hercules graphics adapter modes, and it supports single-scan and multisync monitors as well as the IBM PS/2 analog color and monochrome monitors. SigmaVGA has 256K of on-board memory and permits up to 256 colors to be selected from a palette of 262,144 colors.

According to Sigma Designs, SigmaVGA supports all the new VGA BIOS mode and function calls, including 320 × 200 in 256 colors or 64 gray scales; 640 × 480 in 16 colors or gray scales; 720 × 400 in 16 colors or gray scales; and simultaneous display of two out of a possible eight character sets stored in memory. It can also save and restore video states for use in multitasking applications and offers 132-column support with all digital monitors.

The SigmaVGA graphics board retails for $499 and comes bundled with Show Partner, Version 3.0, from Brightbill-Roberts & Co. of Syracuse, New York. Show Partner is an application designed to capture screens from any source, then present them using a variety of special effects and animation techniques, including color, motion, and sound. Reader Service No. 16.
Sigma Designs Inc.
46501 Landing Pkwy.
Fremont, CA 94538
(415) 770-0100

Video Seven has announced a single-chip EGA that is completely hardware compatible with EGA, CGA, MDA, and HGC (Hercules graphics board) display modes. The chip is contained in a 160-pin package and features a complete implementation of the EGA hardware as well as a fully functional 6845 CRT controller. Advanced features include enhanced automatic switch support, dot clock speeds of up to 34 MHz, double-scan capability, and built-in support logic. The board is manufactured for Video Seven by LSI Logic.

The price has not yet been determined. Reader Service No. 17.
Video-Seven Inc.
46335 Landing Pkwy.
Fremont, CA 94538
(415) 656-7800

Graphics Software Systems has released the OS/2 Graphics Development Toolkit for the advance release of OS/2. The OS/2 GDT includes device drivers for VGA, EGA, and CGA graphics adapters; the Microsoft Mouse; the IBM Proprinter, Graphics Printer, and Color Graphics Printer; the Quietwriter III; and plotters from IBM and HP. Additional drivers now under development include IBM/PS/2 Mouse, the HP LaserJet+, and laser and dot-matrix printers from Epson and other vendors.

The OS/2 GDT advance release includes language binding to support Microsoft C and Macro Assembler. The final version of OS/2 GDT will support OS/2 implementations of C, BASIC, FORTRAN, Pascal, and Macro Assembler from IBM and other third-party vendors.

The list price is $995, with discounts available to registered owners of the GSS Graphics Development Toolkit for DOS. A final version of the OS/2 GDT will be available from GSS when the final version of OS/2 is available from Microsoft and IBM. Reader Service No. 18.
Graphics Software Systems Inc.
9590 S.W. Gemini Dr.
Beaverton, OR 97005
(503) 641-2200

Atron has announced Windows Probe, a set of debugging tools for developers using the Microsoft Windows operating environment. The tools permit trapping programs that are not yet in memory but reside on disk. With many programs operating simultaneously, program bugs have a far greater chance of corrupting other programs as well as themselves. Windows Probe tracks the program in real time and adjusts symbolic and source-level debugging information accordingly.

Atron's AT PROBE contains 1 megabyte of hidden and write-protected memory that stores the Windows Probe debugger software and the symbolic and source-level debugging information without taking memory space in the lower 1 megabyte of system memory. AT PROBE's real-time trace feature lets programmers see how the program operated in real time.

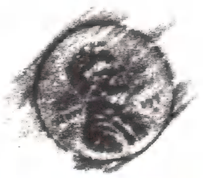
Windows Probe is compatible with the C compiler that is part of the Windows development system. Programmers can display and change local and complex data variables as well as do source-level debugging.

Windows Probe costs $495. Reader Service No. 19.
Atron
20665 Fourth St.
Saratoga, CA 95070
(408) 741-5900

Foresight Resources Corp. has recently released a version of its drafix 1 computer-aided design and drafting program for the Atari ST. Drafix 1/Atari ST is the first professional-quality CAD program available for the Atari 520ST and 1040ST computers. The program is menu-driven with available commands displayed. It includes object drawing and editing, snap grid and object snap drawing aids, multiple fonts, crosshatching, an automatic dimensioning system, and symbol library management.

Drafix 1/Atari ST sells for $195. Reader Service No. 20.
Foresight Resources Corp.
932 Massachusetts
Lawrence, KS 66044
(913) 841-1121

Remember,



Smart Buying Decisions

Start with DDJ

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT #217, CLINTON, IOWA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of Software Tools
FOR THE PROFESSIONAL PROGRAMMER

P.O. Box 2157
Clinton, Iowa 52735-2157

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT #217, CLINTON, IOWA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of Software Tools
FOR THE PROFESSIONAL PROGRAMMER

P.O. Box 2157
Clinton, Iowa 52735-2157

Use this card for FREE, FAST information about the products and services listed in this issue. Simply circle the appropriate numbers below.

Name ______________________________
Title ______________________________
Company ____________________ Phone (____) __________
Address ______________________________
City/State/Zip ______________________________

1	26	51	76	101	126	151	176	201	226	251	276	301	326	351	376
2	27	52	77	102	127	152	177	202	227	252	277	302	327	352	377
3	28	53	78	103	128	153	178	203	228	253	278	303	328	353	378
4	29	54	79	104	129	154	179	204	229	254	279	304	329	354	379
5	30	55	80	105	130	155	180	205	230	255	280	305	330	355	380
6	31	56	81	106	131	156	181	206	231	256	281	306	331	356	381
7	32	57	82	107	132	157	182	207	232	257	282	307	332	357	382
8	33	58	83	108	133	158	183	208	233	258	283	308	333	358	383
9	34	59	84	109	134	159	184	209	234	259	284	309	334	359	384
10	35	60	85	110	135	160	185	210	235	260	285	310	335	360	385
11	36	61	86	111	136	161	186	211	236	261	286	311	336	361	386
12	37	62	87	112	137	162	187	212	237	262	287	312	337	362	387
13	38	63	88	113	138	163	188	213	238	263	288	313	338	363	388
14	39	64	89	114	139	164	189	214	239	264	289	314	339	364	389
15	40	65	90	115	140	165	190	215	240	265	290	315	340	365	390
16	41	66	91	116	141	166	191	216	241	266	291	316	341	366	391
17	42	67	92	117	142	167	192	217	242	267	292	317	342	367	392
18	43	68	93	118	143	168	193	218	243	268	293	318	343	368	393
19	44	69	94	119	144	169	194	219	244	269	294	319	344	369	394
20	45	70	95	120	145	170	195	220	245	270	295	320	345	370	395
21	46	71	96	121	146	171	196	221	246	271	296	321	346	371	396
22	47	72	97	122	147	172	197	222	247	272	297	322	347	372	397
23	48	73	98	123	148	173	198	223	248	273	298	323	348	373	398
24	49	74	99	124	149	174	199	224	249	274	299	324	349	374	399
25	50	75	100	125	150	175	200	225	250	275	300	325	350	375	400

November '87: Use before February 29, 1988

1. Did you buy this issue on a newsstand? (   ) Yes (   ) No
2. Are you a subscriber? (   ) Yes (   ) No
3. Have you purchased a product as a result of seeing it advertised in *Dr. Dobb's Journal?* (   ) Yes (   ) No

Dr. Dobb's Journal of
Software Tools
FOR THE PROFESSIONAL PROGRAMMER

Use this card for FREE, FAST information about the products and services listed in this issue. Simply circle the appropriate numbers below.

Name ______________________________
Title ______________________________
Company ____________________ Phone (____) __________
Address ______________________________
City/State/Zip ______________________________

1	26	51	76	101	126	151	176	201	226	251	276	301	326	351	376
2	27	52	77	102	127	152	177	202	227	252	277	302	327	352	377
3	28	53	78	103	128	153	178	203	228	253	278	303	328	353	378
4	29	54	79	104	129	154	179	204	229	254	279	304	329	354	379
5	30	55	80	105	130	155	180	205	230	255	280	305	330	355	380
6	31	56	81	106	131	156	181	206	231	256	281	306	331	356	381
7	32	57	82	107	132	157	182	207	232	257	282	307	332	357	382
8	33	58	83	108	133	158	183	208	233	258	283	308	333	358	383
9	34	59	84	109	134	159	184	209	234	259	284	309	334	359	384
10	35	60	85	110	135	160	185	210	235	260	285	310	335	360	385
11	36	61	86	111	136	161	186	211	236	261	286	311	336	361	386
12	37	62	87	112	137	162	187	212	237	262	287	312	337	362	387
13	38	63	88	113	138	163	188	213	238	263	288	313	338	363	388
14	39	64	89	114	139	164	189	214	239	264	289	314	339	364	389
15	40	65	90	115	140	165	190	215	240	265	290	315	340	365	390
16	41	66	91	116	141	166	191	216	241	266	291	316	341	366	391
17	42	67	92	117	142	167	192	217	242	267	292	317	342	367	392
18	43	68	93	118	143	168	193	218	243	268	293	318	343	368	393
19	44	69	94	119	144	169	194	219	244	269	294	319	344	369	394
20	45	70	95	120	145	170	195	220	245	270	295	320	345	370	395
21	46	71	96	121	146	171	196	221	246	271	296	321	346	371	396
22	47	72	97	122	147	172	197	222	247	272	297	322	347	372	397
23	48	73	98	123	148	173	198	223	248	273	298	323	348	373	398
24	49	74	99	124	149	174	199	224	249	274	299	324	349	374	399
25	50	75	100	125	150	175	200	225	250	275	300	325	350	375	400

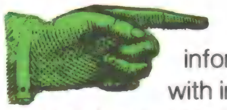
November '87: Use before February 29, 1988

1. Did you buy this issue on a newsstand? (   ) Yes (   ) No
2. Are you a subscriber? (   ) Yes (   ) No
3. Have you purchased a product as a result of seeing it advertised in *Dr. Dobb's Journal?* (   ) Yes (   ) No

Dr. Dobb's Journal of
Software Tools
FOR THE PROFESSIONAL PROGRAMMER

For Free Info ...

Start Here



Smart buyers start with *DDJ*'s free information card, a shopping center filled with information about the products and services advertised in this very issue: everything from software and systems to peripherals and professional support services.

And smart buyers can use this free information card to quickly and easily gather a comprehensive file of facts, figures and product specs to sort out competing claims. Using *DDJ*'s free information card can prevent you from making the wrong, costly buying decision.

Be a smart shopper. Complete and mail this postage paid card today!

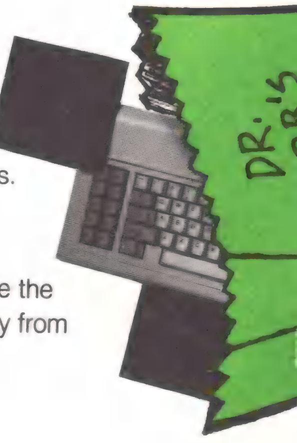
Take a Reader Service Card with You

It's Easy as ...

1. Circle the appropriate free information numbers, referring to the advertiser index for more information.

2. Fill in your name and address.

3. Mail today—postage is absolutely free. You'll receive the product information you need directly from the manufacturers, thanks to *DDJ*.



The Advertiser Index

Advertiser Name	Page #	RS#
Aker Corporation	75	369
Aldebaran Laboratories	25	350
Aspen Scientific	45	360
AT&T Electronic Photography & Imaging	4-5	294
Austin Code Works	97	250
Baysoft	48	383
Blaise Computing	2	159
Borland International	1	161
Breakpoint Computer Systems, Ind.	106	327
Bryte Computer	82	387
Burton Systems Software	80	212
C Users Group	105	181
C Ware	85	*
Cadre Technologies, Inc.	49	261
Centurion Discount Software	99	236
Coder's Source	103	315
Coder's Source	136	152
Compaq Computer Corporation	32A-H	*
Comptech	36	94
Compu View	41	122
Creative Programming	109	*
Crescent Software	148	379
Crosstalk Communications	C6	171
Custom Software Systems	119	268
Datalight	9	203
DDJ Subscriptions	120	*
Desktop A.I.	86	258
Disk Software	102	165
Ecosoft, Inc.	147	89
EKD Computers	14	295
Enertec	80	346
Essential Software	C5	139
Fair-Com	40	93
GenSoft Systems	77	166
Geocomp Corporation	99	111
Gimpel Software	52	*
Gotoless	92	116
Greenleaf Software	125	97
Guidelines Software	65	351
Headlands Corporations	66	354
IMSI	13	293
Komputerwerk	80	388
Lattice, Inc.	113	101
Logic Process Corporation	98	169
Logitech, Inc.	37	257
Lugaru Software Ltd.	65	135
M Street Software	94	275
Machine Independent Software Corp.	79	294
Magma Systems	70	313
Manx Software Systems	7	108
Metagraphics Software Corporation	76	392
MetaWare Incorporated	82	95
Micro Way	122	300
Microprocessors Unlimited	105	105
Microrim	42-43	320
Microsoft	53	*
Microsoft	59	380
Microsoft	61	*
Microsoft	55	*
Microsoft	57	*
Microsoft Press	114	125
Microsoft Press	115	126
MMC AD Systems	123	192
Mortice Kern Systems, Inc.	51	249
Motorola Corporation	10-11	*
Nanosoft Associates	69	309
Nantucket Corporation	27	220
Norton Utilities (The)	45	243
Nostradamus	60	251
NWP Intellegent Solutions, Inc.	121	400

Advertiser Name	Page #	RS#
Oakland Group, Inc. (The)	111	227
Oasys	32	254
Oregon Software	48	357
PC Technologies	21	323
Peacock Systems	107	316
Periscope Co. Inc.	142	214
Pharlap	86	343
PIM Publications	83	136
Pioneering Controls Technologies, Inc.	85	191
PMI	33	239
Polytron Corporation	133	283
PPI	135	390
Programmer's Connection	149-151	129
Programmer's Shop	128-129	133,301,302
Prospero Software	29	160
Prototype	123	*
Quarterdeck Office Supplies	23	284
Raima Corporation	117	*
Rainbow Technologies	93	255
SAS Institute	87	*
Santa Rita Software	15	353
Scantel Systems Limited	92	391
Scientific Endeavors	96	210
Secom Information Products Co.	95	394
Seidl Computer Engineering	35	114
Semaphore, Inc.	105	308
Semi-Disk Systems	96	85
Sharpe Systems Corporation	137	176
Sintar Software	106	*
SLR Systems	146	78
Softfocus	84	259
SoftScience Corporation	139	*
Software Connections Inc.	73	330
Software Garden Inc.	67	314
Software Security, Inc.	50	170
Solution Systems	58	142
Solution Systems	143	296
Tenon Software	83	*
Texas Instruments	16-17	*
Texas Instruments	131	*
Tool Makers (The)	138	319
True Basic	140	344
TSF (The Software Family)	141	230
Turbo Tech Report	127	119
Unipress Software	132	77
Vermont Creative Software	56	157
Wallsoft	68	90
Whitewater Group (The)	71	282
Windsor Technologies	91	*
Zylab	81	329

*This advertiser prefers to be contacted by phone; consult ad.

Advertising Sales Offices

Midwest
Charles Shively (415) 366-3600

Northern California/Northwest
Lisa Boudreau (415) 366-3600

Northeast
Cynthia Zuck (718) 499-9333
Martha Brandt (415) 366-3600

Southern California/AZ/NM/TX
Michael Wiener (415) 366-3600

Director of Marketing and Advertising
Ferris Ferdon (415) 366-3600

Telemarketing Representative/SE/SW USA
Cheri Blum (714) 761-0294

(continued from page 144)

Raster Technologies has introduced two products in a line of GX4000 parallel-processing graphics accelerators for use with Sun Microsystems' workstations. Raster's GX4330 and GX4340 plug directly into a Sun-3 or Sun-4 VME backplane. They use a new parallel architecture that Raster has developed to execute the proposed ANSI PHIGS and PHIGS+ standards at the fastest possible rate.

The systems offer a 24-bit, true-color display and the ability to create and edit a standards-based 3-D display list structure. They support Sun's X.11/NeWS Window System and 200,000 to 1,000,000 transformed and drawn 32-bit floating-point 3-D vectors per second. The architecture consists of a dual-ported Display List Module incorporating 1-megabit, static column RAM chips.

Reader Service No. 21.
Raster Technologies Inc.
Two Robbins Rd.
Westford, MA 01886
(617) 692-7900

MacMemory has released Turbo SE, a 16-MHz 68000-based accelerator board for the Apple Macintosh SE that offers a minimum speed increase of 200 percent. Users also have the option of moving the Macintosh SE ROMs to the Turbo SE board to double the speed of all ROM operations. This increase is 100 percent compatible with existing Mac applications.

Turbo SE sells for $599. Reader Service No. 22.
MacMemory Inc.
2480 N. First St.
San Jose, CA 95131
(408) 922-0140

AJS Publishing has released db/LIB, a database library that features a set of 20 assembly-language procedures that give Microsoft's QuickBASIC (Version 2.0 or later) full relational database management capability and dBASE III standard file compatibility for database, index, and text files.

In addition to the library files, the db/LIB software package contains several executable data management routines, including Browse, Copy Structure, and List. Features include the ability to search and index on memo fields, offer direct access to the keys in index files and to the database file header, and provide access to files down any subdirectory path.

The db/LIB software internally manages its own system of record buffering and, by writing records to memory buffers rather than to disk, makes many applications run appreciably faster. It employs dynamic string allocation for user variables returned by the library, which means that variables do not require preallocation of string space.

Db/LIB sells for $139. Reader Service No. 23.
AJS Publishing
P.O. Box 379
North Hollywood, CA 91603
(818) 985-3383

Sterling Castle Software has just released Version 4.0 of the BlackStar C Function Library. The new version

4 Times Faster

than MASM 5.0

All MASM features (except 386 & CodeView support)

Plus **Generates smaller code! (no inserted NOP's)*
**Automatically handles jumps out of range*
**Handles most of MASM's phase errors*
**Built in MAKE*
**Up to 15,000 symbols*
**Simplified segmentation*

OPTASM
$195

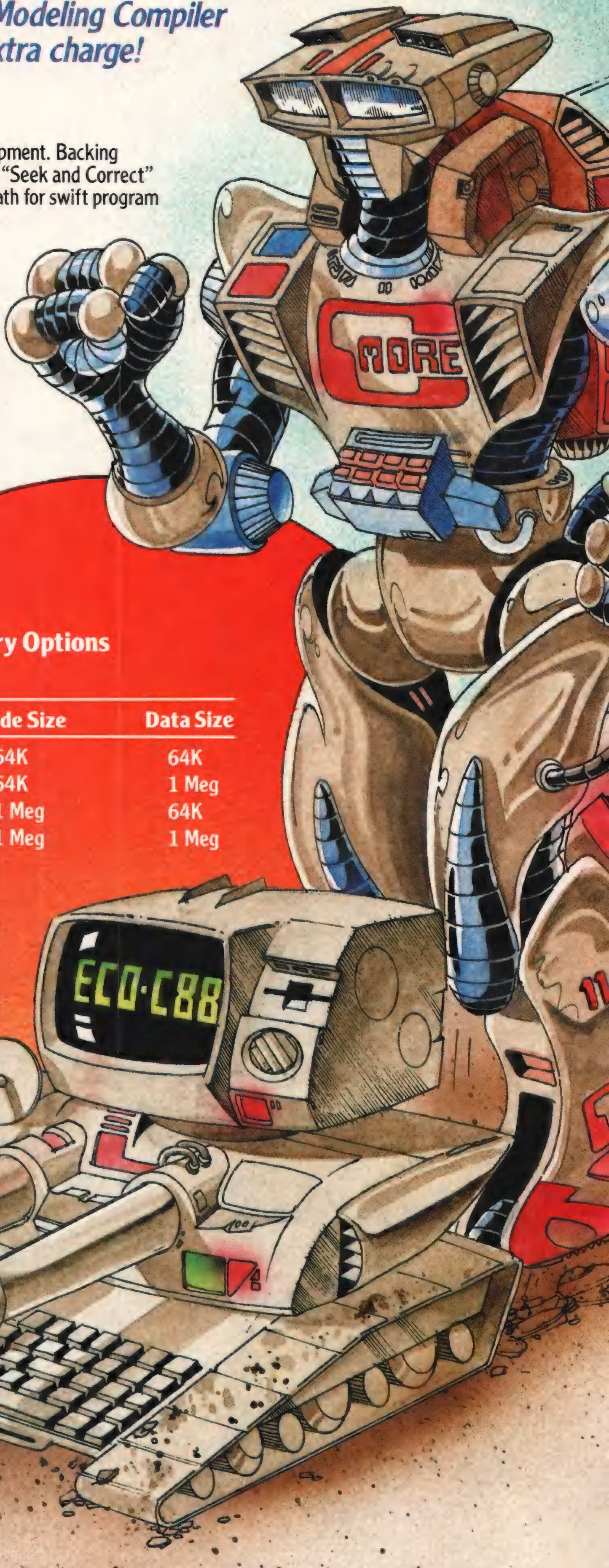
SLR Systems™

1622 N. Main St., Butler, PA 16001
(412) 282-0864 (800) 833-3061
TELEX 559215

CIRCLE 78 ON READER SERVICE CARD

DEBUGGING SWAT TEAM

Order Eco-C88 Rel. 4.0 New Modeling Compiler and get C-more at no extra charge!



Seek and Correct

You already know that fast compilation does not mean fast program development. Backing up for bogus error messages and removing the bugs takes time. Eco-C88's "Seek and Correct" three - way error checking finds even the most elusive bugs, clearing the path for swift program development.

Double Barrel Error Checking

Eco-C88 nails syntax errors cold and tells you about the error in plain English. And there's no avalanche of false error messages, either. Other compilers can generate up to four times the number of error messages actually present; they leave it up to you to guess which ones are real. You'll be more productive with Eco-C88 because there is no guess work.

Eco-C88 provides ten levels of semantic error checking. You can select from almost no checking to the fussiest you've ever seen. Eco-C88's "picky flag" finds subtle errors that slip by other compilers.

Eco-C88 also features:

- All data types, plus ANSI Enhancements
- Robust library, including many new ANSI functions
- CED editor with online function help, split windows, compile-edit-link capability
- New, expanded manual with sample programs for the library functions

Memory Options

Model	Code Size	Data Size
Small	64K	64K
Compact	64K	1 Meg
Medium	1 Meg	64K
Large	1 Meg	1 Meg

C-more Source Code Debugger

Finally, if a really nasty bug persists, put C-more, our source code debugger, to work. With C-more you can watch your program as it executes, single-step it, set simple or conditional breakpoints, test complex expressions, use variables as indexes into other variables, initialize and trace variables, examine CPU registers, display results with printf()-type options and much more. C-more can help you track down bugs in minutes rather than days.

The price for Eco-C88 is $99.95. And, for a limited time, we'll give you our C-more debugger at no extra charge.

Ecosoft Inc.
6413 N. College Ave.
Indianapolis, IN 46220

(317) 255-6476 (Tech Info)
(800) 952-0472 (Orders)

(continued from page 146)

supports the EGA, allows terminate-and-stay-resident program development, and has six new serial communication functions. The TSR function enables the development of programs that can pop up and be utilized while another program is running. The library supports the standard ANSI language; is compatible with Microsoft C, Version 3.0/4.0, and Lattice C, Version 3.0; and is adaptable to other versions.

The product offers more than 300 functions, including device handlers for screen, graphics, keyboard, printer, and mouse. It also has the capabilities of interrupts, string, menu, date, time, and system functions and uses primitive functions written in assembly language to improve speed and memory usage. The library includes complete source code and small-, medium-, and large-memory models.

Version 4.0 costs $129. Reader Service No. 24.
Sterling Castle Software
702 Washington St., Ste. 174
Marina del Rey, CA 90292
(213) 306-3020

Silicon Beach Software has introduced Super 3D for the Macintosh, a product that offers 3-D graphics modeling and animation for professional engineering and graphics arts applications. It was designed to provide artists, architects, engineers with dynamic simulation and modeling capabilities.

Modeling tools allow users to create shaded shapes in more than 16,000 colors, 3-D animation for dynamic visualization of complex structures, and movie-camera-like tools for precise visual control. The tool palette offers ten basic tools for creating graphics primitives such as points, lines, arcs, circles, ovals, rectangles and polygons. Objects can revolve about any axis or they can be extruded, translated, scaled, flipped, replicated, and mirrored or arbitarily reshaped.

Super 3D is priced at $295. Reader Service No. 25.
Silicon Beach Software Inc.
P.O. Box 261430
San Diego, CA 92126
(619) 695-6956

NeXT and Adobe Systems Inc. have announced a jointly developed version of PostScript for workstation displays. The new product, Display PostScript, will be independent of windowing systems and include full support for outline fonts, arbitrary line-widths, rotation, and color.

According to Steve Jobs, "Display Postscript allows us to achieve true WYSIWYG from the display to the printed page since the same imaging model is now used for both."

No price has been announced because the product is still under development. Adobe has scheduled a demonstration of the product for the summer of 1988. Circle Reader Service No. 26.
NeXT, Inc.
3475 Deer Creek Road
Palo Alto, CA 94304
(415) 424-0200

DDJ

BASIC Programming Tools

There's nothing basic about these professional programming utilities.

Whether you're a seasoned expert or just starting out, we can help you create programs that run faster, work harder, and simply look better. We have built our reputation on customer satisfaction by providing expert advice and quality technical support. All Crescent Software products include source code, demonstration programs, clear documentation, and a 30-day satisfaction guarantee.

- **QBase** is a superb screen designer *and* full-featured relational database. Because we include complete BASIC source code, QBase can be customized to suit your needs. Besides its database capabilities, QBase is outstanding for creating custom titles, help screens, and product demos, and includes a versatile slide-show program. **$99**

NEW!
- **QBase Report** enhances QBase by generating reports with multiple levels of sorting, automatic record updating, browsing, and transaction posting. As a special bonus, QBase Report includes a complete, ready to run, time billing application for computer professionals. **$69**
- **QuickPak** contains more than 65 essential routines for BASIC programmers. Included are programs for windowing, access to DOS and BIOS services, searching and sorting string arrays, creating pull-down and Lotus™ menus, accepting data input, and much more. QuickPak is loaded with examples and tutorial information, and comes with a clever tips and tricks book, plus *The Assembly Tutor* — a complete guide to learning assembly language from a BASIC perspective. **$69**
- **By Customer Demand — QuickPak II** More than 30 additional tools, including disk and printer tests to eliminate the need for On Error in your programs, and a multi-line text input routine that lets you put a note pad with full word wrap anywhere on the screen from within your programs. Other routines include binary file access, more menus, multiple screen save and restore, continuous time display, automatic box drawing, and much more. (Available for use with QuickBASIC *only.*) **$49**

NEW!
- **GraphPak** is an extensive collection of routines for displaying line, bar, and pie charts automatically within your programs. It will create 3-D charts with manual or automatic scaling, titles and legends in any size or style, as well as scrolling and windowing *in graphics*. GraphPak also comes with a sophisticated font editor for customizing your own character sets. **$69**

CRESCENT SOFTWARE

64 Fort Point Street, East Norwalk, CT 06855
(203) 846-2500

Separate versions are available for Microsoft QuickBASIC and Borland Turbo Basic — *please* specify when ordering. No royalties, not copy protected, of course. We accept Visa, M/C, C.O.D. and Checks. Add $3 shipping and handling, $10 overnight and foreign, $25 2nd day foreign.

CIRCLE 379 ON READER SERVICE CARD

Upgrade your technology

The software technology available to programmers of IBMcompatible personal computers is truly amazing. And newer, more powerful development packages appear all the time. But until now, finding out about these important products has been a difficult and time consuming task.

Large Inventory. We have one of the largest inventories of programmer's development products in the industry. Most orders are shipped within 24 hours.

Noncommissioned Staff. Our courteous salespeople are always ready to help you. And if you aren't sure about your needs, our knowledgeable technical people can give you sound, objective advice.

Experience. We've specialized in development software for IBMcompatible personal computers since 1984 and are experienced at providing a full range of quality products and customer services.

FREE Buyer's Guide. The New 76 page Programmer's Connection Fall 1987 Buyers Guide contains individual descriptions of over 500 titles of programmer's development software by over 150 manufacturers. Each description covers major product features as well as any software or hardware requirements and version numbers. In the box on the right are some examples of the types of descriptions you'll find in our Buyer's Guide.

No Hidden Charges. The low discount prices in our Buyer's Guide are all you pay. We don't charge extra for UPS Ground shipping, credit cards, COD orders, purchase orders, sales tax (except Ohio) or special handling (except for non-Canadian international orders).

Guarantees. We offer FREE 30-day no-risk return guarantees and 30-day evaluation periods on most of our products.

Latest Versions. The products we carry are the latest versions and come with the same manufacturer's technical support as if buying direct.

Blaise
C TOOLS PLUS/5.0
List $129 Ours $99

C TOOLS PLUS/5.0 is a library for Microsoft C that can provide you with a full spectrum of general-purpose utility functions. Included are functions that take advantage of the machine features of IBM-compatible personal computers and DOS and complement the standard compiler library. Almost all functions are written in C using techniques most suitable for good C program design. Some of the areas covered are: extensive string handling; screen handling including support for multiple monitors and the EGA; general utility and keyboard functions; DOS memory management; windows that can be stacked, removed and accept user input; intervention code; and interrupt service routine support for truly flexible resident applications. C TOOLS PLUS/5.0 includes all source code, complete examples and a comprehensive reference manual.

Supports Microsoft C 5.0 and Microsoft QuickC.

ESP
Command Plus
List $80 Ours $69

Command Plus is a powerful, bootable MS-DOS command processor fully compatible with COMMAND.COM. Many new features are included that will help you increase your speed, productivity and ease of programming. These include a history processor that lets you recall and edit previously entered commands using the cursor keys. The alias facility allows the creation of command macros with replaceable parameters. Other features include: regular expressions in filenames; directory and argument stacks; a command line editor; access to environment variables; a MOVE command; and Browse, a full screen file viewer. Command Plus also includes SCRIPT, a batch processor that uses a Pascal-like language that features: integer and string variables; boolean, math and string operators; CALL, FOR/WHILE, GOTO, IF/THEN/ELSE, and SWITCH statements; and display and file access routines. Enhancements to DIR include sort options and file attrib display. COPY options include selection by date/time range, and recursive sub-directory processing.

Requires 48K memory. Version 1.2.

Meridian Software
AdaVantage Compiler 2.0
List $795 Ours $735

The Meridian AdaVantage Compiler is a fully validated implementation of the Ada language. The compiler generates native 8086 code in the Intel standard object format. A linker is provided to create stand-alone executable files for MS-DOS. The Meridian AdaVantage Compiler package includes the compiler, linker, library management tools, support packages, runtime libraries and a configuration tool. In addition to the standard Ada packages, support packages are provided to make integer, floating point and text I/O more convenient to implement. The compiler also provides a pragma INTERFACE used to make calls to subprograms written in 8086 assembly language, Meridian C or Meridian Pascal. A source level debugger will be available in late 1987.

Requires hard disk and 640K memory. Runtime fees apply if more than 99 copies are sold. Version 2.0.

How to Get Your Copy. There are three ways for you to receive your FREE copy of the Programmer's Connection Buyer's Guide: 1) Use the reader service card provided by this journal; 2) Mail us a card or letter with your name and address; or 3) Call one of our convenient toll free telephone numbers.

If you haven't yet received your Programmer's Connection Fall 1987 Buyer's Guide, act now. Upgrading your programming technology could be one of the wisest and most profitable decisions you'll ever make.

CALL TOLL FREE

USA: 800-336-1166
Canada: ... 800-225-1166
Ohio & Alaska (Collect): . . 216-494-3781
International: 216-494-3781
Telex: 9102406879
Easylink: 62806530

Programmer's Connection
7249 Whipple Ave. N.W.
North Canton, OH 44720

Please turn the page for our latest price list and ordering information.

PROGRAMMER'S CONNECTION
QUALITY INTEGRITY Service SUPPORT PRICE

The Free Programmer's Connection Fall '87 Buyer's Guide.

CIRCLE 129 ON READER SERVICE CARD

ai - expert systems

	List	Ours
1st-CLASS by Programs in Motion	495	399
EXSYS Development Software by EXSYS	395	309
EXSYS Runtime System	600	469
LEVEL5 by Information Builders New	685	569
Logic-Line Series All varieties by Thunderstone	CALL	CALL

ai - lisp language

	List	Ours
Golden Common LISP by Gold Hill	495	CALL
Golden Common LISP Developer by Gold Hill	1190	CALL
Q'Nial Various by NIAL Systems	CALL	CALL
Star Sapphire LISP Compiler by Sapiens New	495	429
TransLISP PLUS from Solution Systems	195	125

ai - prolog language

	List	Ours
Arity Combination Package	1095	979
Expert System Development Pkg	295	229
File Interchange Toolkit	50	44
PROLOG Compiler & Interpreter	650	569
Screen Design Toolkit	50	44
SQL Development Package	295	229
Arity PROLOG Interpreter	295	229
Arity Standard Prolog	95	77
LPA microPROLOG All Varieties	CALL	CALL
MPROLOG Language Primer LOGICWARE	50	45
MPROLOG P500 by LOGICWARE	495	395
MPROLOG P550 w/Primer by LOGICWARE	220	175
Turbo PROLOG by Borland Intl	100	64
Turbo PROLOG Toolbox by Borland Intl	100	64

ai - smalltalk language

	List	Ours
Smalltalk/V	100	84
EGA/VGA Color Option	50	45
Goodies Diskette	50	45
Smalltalk/Comm	50	45

ai - texas instruments

	List	Ours
Arborist Decision Tree Software	595	519
PC Scheme Lisp New Version	95	84
Personal Consultant Easy New Version	495	435
Personal Consultant Image	495	435
Personal Consultant Online	995	869
Personal Consultant Plus New Version	2950	2589
Personal Consultant Runtime	95	84

ada language

	List	Ours
AdaVantage GSA-Validated by Meridian Software	795	735
AdaVantage Utility Packages	50	47
DOS Environment Package	50	47
Janus/ADA C Pak by R&R Software	95	84
Janus/ADA D Pak by R&R Software	1250	1059
Janus/ADA ED Pak by R&R Software	395	349

apl language

	List	Ours
APL*PLUS PC by STSC	695	495
APL*PLUS PC Spreadsheet Mgr by STSC	195	139
APL*PLUS PC Tools Vol 1 by STSC	295	199
APL*PLUS PC Tools Vol 2 by STSC	85	58
APL*PLUS PS/2 by STSC New	695	495
ATLAS*GRAPHICS by STSC	450	329
Financial/Statistical Library by STSC	275	189
Pocket APL by STSC	95	69
STATGRAPHICS by STSC	895	649

assembly language

	List	Ours
386 ASM/LINK Cross Asm by Phar Lap	495	389
8088 Assembler w/Z-80 Translator by 2500 AD	100	89
ASMLIB Function Library by BCSoft	149	125
asmTREE B-Tree Dev System by BCSoft	395	329
Cross Assemblers Various by 2500 AD	CALL	CALL
EZASM by C Source	70	59
Microsoft Macro Assembler	150	93
Turbo Debugger by Speedware	89	79
Turbo Editasm by Speedware	99	84
Visible Computer: 8088 by Software Masters	80	64

basic language

	List	Ours
db/Lib for QuickBASIC by AJS Publishing	139	119
Finally by Komputerwerk	99	85
MACH 2 by Micro Help	69	55
Microsoft QuickBASIC	99	63
QBase Relational Database by Crescent	89	79
Quick-Tools by BCSoft	130	109
QuickPak by Crescent Software	69	59
Scientific Subroutine Library by Peerless	125	99
Screen Sculptor by Software Bottling	125	91
Stay-Res by MicroHelp	69	55
True Basic w/Run-time Special Price	200	99
True Basic	100	79
Run-time Module	100	79
Various Utilities	50	41
Turbo BASIC Compiler by Borland Intl	100	64

blaise products

	List	Ours
ASYNCH MANAGER Specify C or Pascal	175	135
C TOOLS PLUS/5.0 New Version	129	99
KeyPlayer Super Batch Program New	50	45
LIGHT TOOLS for Datalight C	100	65
PASCAL TOOLS	125	95
PASCAL TOOLS 2	100	79
PASCAL TOOLS & TOOLS 2	175	135
RUNOFF Text Formatter	50	45
TURBO ASYNCH PLUS	100	79
TURBO C TOOLS	129	99
TURBO POWER TOOLS PLUS	100	79
VIEW MANAGER Specify C or Pascal	275	199

borland products

	List	Ours
EUREKA Equation Solver	167	105
Reflex: The Analyst	150	99
Sidekick	85	57
Superkey	100	64
Turbo Basic Compiler	100	64
Turbo Basic Database Toolbox	100	64
Turbo Basic Editor Toolbox	100	64
Turbo Basic Telecom Toolbox	100	64
Turbo C Compiler (Call for support products)	100	64
Turbo Lightning and Word Wizard	150	94
Turbo Lightning	100	64
Turbo Lightning Word Wizard	70	47
Turbo Pascal and Tutor New Version	CALL	CALL
Turbo Pascal New Version	100	64
Turbo Pascal Tutor New Version	70	41
Turbo Pascal Database Toolbox New Version	100	64
Turbo Pascal Editor Toolbox New Version	100	64
Turbo Pascal Gameworks Toolbox New Version	100	64
Turbo Pascal Graphix Toolbox New Version	100	64
Turbo Pascal Numerical Methods Toolbox New	100	64
Turbo Prolog Compiler	100	64
Turbo Prolog Toolbox	100	64

c compilers

	List	Ours
C86PLUS by Computer Innovations	497	359
DeSmet C w/Debugger & Large case	209	184
DeSmet C w/Debugger only	159	138
Eco-C Complete System by Ecosoft	140	119
Instant C by Rational Systems	495	369
Instant-C/16M by Rational Systems New	895	695
Lattice C Compiler vers. 3.2 from Lattice	500	265
Mark Williams Let's C w/csd	75	54
Microsoft C Compiler w/CodeView New Version	450	269
Microsoft QuickC Compiler New	99	63
Optimum-C by Datalight	139	95
Turbo C Compiler by Borland	100	64
Uniware 68000/10/20 Cross Compiler by SDS	995	829

c interpreters

	List	Ours
C-terp by Gimpel, Specify compiler	298	219
C Trainer with Book by Catalytix	122	87
Introducing C by Computer Innovations	125	99
Run/C by Age of Reason	120	69
Run/C Professional by Age of Reason	250	145

c utilities

	List	Ours
Blackstar C Library by Sterling Castle New	125	98
C++ by Guidelines w/version 1.1 kernel	195	172
c-tree & r-tree Combo by FairCom	650	519
c-tree ISAM File Manager	395	315
r-tree Report Generator	295	239
Csharp Realtime Toolkit by Systems Guild	600	489
Curses Window Dev Pkg by Aspen Scientific New	119	105
with Source Code	289	249
dBx dBASE to C Translator by Desktop AI	350	299
with Source Code	550	419
Flash-up Windows by Software Bottling	90	78
GraphiC Color version by Sci Endeavors	350	274
GRAFLIB by Sutrasoft	175	159
HALO Graphics by Media Cybernetics	300	205
HALO Development Pkg for Microsoft	595	389
The HAMMER by OES Systems	195	129
PANEL Forms Management by Roundhill	295	215
PANEL/TC for Turbo C by Roundhill	129	95
PANEL Plus by Roundhill	495	395
PC Lint by Gimpel Software	139	99
PLOTHP by Sutrasoft	175	159
RTC PLUS Fortran to C by Cobalt Blue	450	399
Sapiens V8 Virtual Memory Manager New	300	265
Scientific Subroutine Library by Peerless	175	135
TE Text Editor source by Sub Systems	95	85
Vitamin C by Creative Programming	225	149
VC Screen Forms Designer	100	79
Zview by Data Mgmt Consultants	245	139

cobol language

	List	Ours
COBOLspII by Flexus	395	329
FPLIB for Realia COBOL by BCSoft	149	129
Micro Focus COBOL See Micro Focus Section		
Microsoft COBOL See Microsoft Section		
PCDT by Pro-Code	995	895
Realia COBOL with RealMENU	1145	899
Realia COBOL	995	783
RealCICS	995	783
RM/COBOL by Ryan-McFarland	950	639
RM/COBOL 85 by Ryan-McFarland	1250	895
RM/NET+5 by Ryan-McFarland New	300	259
RM/Screens New	395	334
SCREENIO by Norcom	400	379
screenplay for COBOL by Flexus	175	129

css products

	List	Ours
Combo Package by Custom Software Systems	199	175
PC/SPELL Spelling Checker	49	45
PC/TOOLS UNIX-like Utilities	49	45
PC/VI vi Editor	149	99

debuggers & profilers

	List	Ours
386 DEBUG Cross Debugger by Phar Lap	195	129
Advanced Trace-86 by Morgan Computing	175	115
Codesmith-86 by Visual Age	145	98
DSD87 by Soft Advances	125	79
MiniProbe by Atron	395	369
Periscope I with Board by Periscope	345	275
Periscope II with NMI Breakout Switch	175	139
Periscope II-X Software only	145	105
Periscope III 8 MHz version	995	795
Periscope III 10 MHz version	1095	875
The PROFILER with Source Code by DWB	125	89
TURBOsmith Source debugger for Turbo Pascal	99	89
The WATCHER Profiler by Stony Brook	60	51

disk utilities

	List	Ours
Back-It by Gazelle Systems New Version	130	115
Disk Optimizer by Softlogic Systems	60	55
Disk Technician by Prime Solutions New	100	89
FASTBACK by 5th Generation Systems	179	129
Vcache by Golden Bow Systems	50	47
Vopt by Golden Bow Systems	50	47
Vfeature by Golden Bow Systems	80	74
Vfeature Deluxe by Golden Bow Systems	120	111
XenoCopy-PC by XenoSoft	80	69

dos utilities

	List	Ours
Advanced Norton Utilities	150	99
Command Plus by ESP Software	80	69
Desqview from Quarterdeck	100	85
FANSI-CONSOLE by Hersey Micro	75	62
Mace Utilities Paul Mace Software New	99	89
MicroHelp Utility by MicroHelp	59	49
Norton Commander by Peter Norton	75	55
Norton Utilities by Peter Norton	100	59
OPAL Shell Language by Software Factory	99	89
Q-DOS II by Gazelle Systems	70	59
Taskview by Sunny Hill Software	80	55

essential products

	List	Ours
C Utility Library	185	119
Essential Comm Library with Debugger	250	189
Essential Comm Library Software Only	185	125
Breakout Debugger Only Any language	125	89
Essential Graphics	250	183

forth language

	List	Ours
CFORTH Native Code Compiler by LMI	300	229
FORTH/83 Metacompiler Specify Target	750	599
PC/FORTH by Laboratory Microsystems	150	109
PC/FORTH+ by Laboratory Microsystems	250	199
Programmer's Package #1 by LMI New	250	199
Programmer's Package #2 by LMI New	350	279
Programmer's Package #3 by LMI New	500	399
UR/FORTH Also Available for OS/2 by LMI	350	279
UR/FORTH Libraries	500	395

fortran language

	List	Ours
50 MORE: FORTRAN by Peerless Scientific	125	95
ACS Time Series by Alpha Computer Service	495	389
AUTOMATED PROGRAMMER by KGK Automated	995	949
Essential Graphics by Essential Software	250	183
Forlib-Plus by Alpha Computer Service	70	44
FORTLIB by Sutrasoft	125	109
FORTRAN Addendum by Impulse Engr	95	85
FORTRAN Addenda by Impulse Engr	165	138
GRAFLIB by Sutrasoft	175	159
HALO Graphics by Media Cybernetics	300	205
I/O PRO w/No Limit Library by MEF	250	219
Microcompatibles Combo Package	240	215
Grafmatic	135	117
Plotmatic	135	117
Microsoft FORTRAN w/CodeView	450	269
No Limit Library by MEF Environmental	129	109
Numerical Analyst by MAGUS	295	249
PANEL by Roundhill Computer Systems	295	215
PLOTHP by Sutrasoft	175	159
RM/FORTRAN by Ryan-McFarland New Version	599	399
RTC PLUS Fortran to C by Cobalt Blue	450	399
Scientific Subroutine Lib by Peerless	175	135
Statistician by Alpha Computer Service	295	235
STATLIB.GL: by Peerless	295	239
STATLIB.TSF: by Peerless	295	239
Strings & Things by Alpha Computer Service	70	45

greenleaf products

	List	Ours
Greenleaf C Sampler for Turbo C & QuickC New	95	69
Greenleaf Comm Library	185	125
Greenleaf Data Windows Library	225	155
with Source Code	395	249
Greenleaf Functions	185	125

help utilities

	List	Ours
HELP/Control by MDS	125	99
On-line Help from Opt-Tech	149	99
SoftScreen/HELP by Dialectic Systems	195	149

lattice products

	List	Ours
Lattice C Compiler ver 3.2 from Lattice	500	265
with Library Source Code	900	495
C Cross Reference Generator	50	37
with Source Code	200	139
C-Food Smorgasbord Function Library	150	95
with Source Code	300	179
C-Sprite Source Level Debugger Special Offer	175	CALL
Curses Screen Manager	125	85
with Source Code	250	169
dBC III	250	169
with Source Code	500	356
dBC III Plus	750	594
with Source Code	1500	1184
LMK Make Facility	195	138
RPG II Combo All three items below	1100	875
RPG II Compiler No Royalties	750	625
SEU Source Entry Utility	250	199
Sort/Merge	250	199
Screen Design Aid Utility for RPG II	350	309
SecretDisk II Encryption Utility New	120	88
SideTalk Resident Communications	120	88
SSP/PC Scientific Subroutine Library	350	269
Text Management Utilities	120	88

metagraphics products

	List	Ours
LightWINDOW/C for Datalight C	95	79
FontWINDOW	95	79
FontWINDOW/PLUS	275	229
MetaWINDOW No Royalties	195	159
MetaWINDOW/PLUS	275	229
TurboWINDOW/C for Turbo C	95	79
TurboWINDOW/Pascal for Turbo Pascal	95	79

micro focus products

	List	Ours
Micro Focus COBOL/2	900	729
Micro Focus Level II COBOL w/Animator	495	395
Level II COBOL	349	279
Level II Animator	195	155
Micro Focus Level II COBOL/ET for UNIX	CALL	CALL
Micro Focus PC-CICS New	1495	1189
with Micro/SPF New	1595	1269
Micro Focus Personal COBOL	149	119
Micro Focus Professional COBOL	2000	1595
Micro Focus VS COBOL/XENIX	1495	1195
Micro Focus Support Products:		
COBOL/IQ Ad hoc Report Writer	495	395
COBOL/IQ for DOS 3.X Networks	995	795
FORMS-2	295	235
SOURCEWRITER	995	795

microport products

386 Unlimited License Kit	249	209
AT Unlimited License Kit	249	209
DOSMerge286 *Run DOS and UNIX together*	149	129
DOSMerge386 *Run DOS and UNIX together*	CALL	CALL
System V/386 Combination	799	699
386 Runtime System	199	169
386 Software Development System	499	429
Text Preparation System	199	169
System V/AT Combination	549	465
AT Runtime System	199	169
AT Software Development System	249	209
Text Preparation System	199	169

microsoft products

Microsoft BASIC Compiler *for XENIX*	695	419
Microsoft BASIC Interpreter *for XENIX*	350	209
Microsoft C Compiler *w/CodeView* .... *New Version*	450	269
Microsoft COBOL Compiler *with COBOL Tools*	700	429
for XENIX	995	609
Microsoft FORTRAN *Optimizing Compiler/CodeView*	450	269
Microsoft FORTRAN *for XENIX*	695	419
Microsoft Learning DOS	50	36
Microsoft MACH 10 *with Mouse & Windows*	549	369
Microsoft MACH 10 *Board only*	399	279
Microsoft Macro Assembler .... *New Version*	150	93
Microsoft Mouse *for IBM PS/2*	CALL	CALL
Microsoft Mouse *Bus Version*	CALL	CALL
Microsoft Mouse *Serial Version*	CALL	CALL
Microsoft Pascal Compiler	300	179
for XENIX	695	419
Microsoft QuickBASIC	99	63
Microsoft QuickC .... *New*	99	63
Microsoft Windows	99	63
Microsoft Windows Development Kit	500	299
Microsoft Word .... *New Version*	450	269

mks products

MKS AWK .... *New*	75	65
MKS RCS *Revision Control System* .... *New*	189	155
MKS Toolkit *with MKS VI Editor*	139	109
MKS Trilogy *with AWK, CRYPT & Korn Shell* ... *New*	119	99
MKS VI *Editor by MKS* .... *New*	75	65

modula-2 language

LOGITECH Modula-2 Development System .... *New*	249	199
Modula-2 Compiler Pack .... *New*	99	79
Modula-2 Toolkit .... *New*	169	139
LOGITECH Modula-2 ROM Pkg	299	239
LOGITECH Modula-2 Window Pkg	49	39
Macro2 *Macro preprocessor by PMI*	89	79
ModBase *by PMI*	89	79
ModGraph *by TEQNA* .... *New*	50	45
Repertoire *by PMI*	89	75
Science & Engrg Tools *by Quinn-Curtis*	75	67
Universal Graphics Library *by Quinn-Curtis*	130	119

mouse products

LOGIMOUSE BUS *with PLUS Pkg by LOGITECH*	119	98
with PLUS & PC Paintbrush	149	119
with PLUS & CAD Software	189	153
with PLUS & CAD & Paint	219	179
with First Publisher .... *New*	CALL	CALL
LOGIMOUSE C7 *with PLUS Pkg, Specify Connector*	119	98
with PLUS & PC Paintbrush	149	119
with PLUS & CAD Software	189	153
with PLUS & CAD & Paint	219	179
with First Publisher .... *New*	CALL	CALL
Microsoft Mouse *See Microsoft Section*		

other languages

ACTOR *by Whitewater Group* .... *New Version*	495	419
CCS MUMPS *Single-User by MGlobal*	60	50
CCS MUMPS *Single-User Multi-Tasking*	150	129
CCS MUMPS *Multi-User*	450	359
Marshal Pascal *by Marshal Language Systems*	189	165
Pascal-2 *by Oregon Software*	395	325
Personal REXX *by Mansfield Software*	125	99
SNOBOL4+ *by Catspaw*	95	80

other products

Carbon Copy Plus *by Meridian Technology*	195	159
Dan Bricklin's Demo Pgm *by Software Garden*	75	57
Dan Bricklin's Demo Tutorial	50	45
Fast Forward *by Mark Williams*	70	59
Instant Replay *by Nostradamus*	150	CALL
MicroTEX *Typesetting from Addison-Wesley*	295	CALL
Printer Drivers .... *New*	CALL	CALL
Net-Tools *by BCSoft*	149	129
Norton Guides *Specify Language* .... *New*	100	75
OPT-Tech Sort *by Opt-Tech Data Proc*	149	99
PC/TOOLS *by Custom Software*	49	45
Screen Machine *by MicroHelp*	79	59
SuperSort *by LifeStyle* .... *New*	139	119

phoenix products

Pasm86 *Macro Assembler version 2.0*	195	108
Pdisk *Hard Disk & Backup Utility*	145	99
Pfantasy Pac *Phoenix Combo*	995	595
Pfinish *Execution Profiler*	395	209
Pfix86plus *Symbolic Debugger*	395	209
PforCe *Specify C Compiler*	395	209
PforCe++ *Specify C Compiler and C++*	395	209
Plink86plus *Overlay Linker*	495	275
Pmaker *Make Utility*	125	78
Pmate *Macro Text Editor*	195	108
Pre-C *Lint Utility*	295	154
Ptel *Binary File Transfer Program*	195	108

polytron products

PolyBoost *Software Accelerator* ..... *Special Price*	80	54
PolyDesk III	99	72
PolyDesk III Archivist	50	42
PolyDesk III Cryptographer	50	42
PolyDesk III Talk	70	52
PolyLibrarian *Library Manager*	99	89
PolyLibrarian II *Library Manager*	149	129
PolyMake *UNIX-like Make Facility*	149	129
PolyShell .... *Special Price*	149	105
Polytron C Beautifier	50	45
PolyXREF *Complete Cross Ref Utility*	219	185
PolyXREF *One language only*	129	109
PVCS Corporate *Version Control System*	395	329
PVCS Personal	149	129

program mgmt utilities

Interactive EASYFLOW *by Haventree* .. *New Version*	150	125
PrintQ *by Software Directions*	89	84
Quilt Computing Combo Package	250	199
QMake *Program Rebuild Utility*	99	79
SRMS *Software Revision Mgmt System*	185	159
Sapiens MAKE & V8 .... *New*	439	379
Sapiens MAKE .... *New*	179	155
Source Print *by Aldebaran Labs*	97	75
TLIB *Version Control System by Burton*	100	89
Tree Diagrammer *by Aldebaran Labs*	77	67

raima products

dbQUERY *Single-User Query Utility*	195	129
Single-User with Source Code	495	389
Multi-User	495	389
Multi-User with Source Code	990	699
dbVISTA *Single-User DBMS*	195	129
Single-User with Source Code	495	389
Multi-User	495	389
Multi-User with Source Code	990	699

sco products

Complete XENIX System V *by SCO*	1295	994
Development System	595	499
Operating System *Specify XT or AT*	595	499
Text Processing Package	195	144
Lyrix *by SCO*	595	449
SCO Professional *1-2-3 Workalike for XENIX*	795	595
SCO XENIX-NET	595	495
XENIX System V 386 *by SCO* .... *New*	CALL	CALL

softcraft products

Btrieve *ISAM Mgr with No Royalties*	245	184
Xtrieve *Query Utility*	245	184
Report Option *for Xtrieve*	145	99
Btrieve/N *for Networks*	595	454
Xtrieve/N	595	454
Report Option/N *for Xtrieve/N*	345	269

text editors

Brief & dBrief Combo *from Solution Systems*	275	CALL
Brief	195	CALL
dBrief *Customizes Brief for dBASE III*	95	CALL
de *by David Livshin* .... *New*	75	65
Epsilon *Emacs-like editor by Lugaru*	195	147
KEDIT *by Mansfield Software*	125	98
Micro/SPF *by PHASER SYSTEMS*	175	139
Microsoft Word .... *New Version*	450	269
PC/VI *by Custom Software Systems*	149	99
SPF/PC *by Command Technology Corp*	CALL	CALL
Vedit Plus *by CompuView*	185	128

turbo pascal utilities

ALICE *Interpreter by Software Channels*	95	66
DOS/BIOS & Mouse Tools *by Quinn-Curtis*	75	67
Flash-up Windows *by Software Bottling*	90	78
MACH 2 *for Turbo Pascal by Micro Help*	69	55
MetraByte D/A Tools *by Quinn-Curtis*	100	89
Science & Engrg Tools *by Quinn-Curtis*	75	67
Screen Sculptor *by Software Bottling*	125	91
Speed Screen *by Software Bottling*	35	32
System Builder *by Royal American* .... *New Version*	150	129
IMPEX *Query Utility*	100	89
Report Builder .... *New Version*	130	115
TDebugPLUS *by TurboPower Software*	60	49
Tmark *by Tangent Designs*	80	69
Turbo EXTENDER *by TurboPower Software*	85	64
Turbo OPTIMIZER *by TurboPower*	75	65
with Source Code	125	98
Turbo Professional *by Sunny Hill*	70	45
TurboHALO *from IMSI*	129	98
TurboPower Utilities *by TurboPower*	95	78
TurboRef *by Gracon Services*	50	35
TURBOsmith *Source Debugger by Visual Age*	99	89
Universal Graphics Library *by Quinn-Curtis*	130	119

wendin products

Operating System Toolbox	99	79
PCNX *Operating system*	99	79
PCVMS *Similar to VAX/VMS*	99	79
Wendin-DOS *Multitasking DOS*	99	85
Wendin-DOS Application Developer's Kit ..... *New*	99	85
XTC *Text Editor w/Pascal source*	99	75

xenix/unix products

Btrieve *ISAM File Mgr by SoftCraft*	595	454
C-terp *by Gimpel, Specify compiler*	498	379
c-tree *ISAM Mgr by FairCom*	395	315
dBx *with Library Source by Desktop AI*	550	419
DIRECTORY SHELL 286 *by American Mgmt Sys* .... *New*	349	295
DIRECTORY SHELL 386 *by American Mgmt Sys* .... *New*	495	415
DOSIX *Console Version by Data Basics*	400	349
DOSIX *User Version by Data Basics*	200	179
Micro Focus Products *See Micro Focus Section*		
Microport Products *See Microport Section*		
Microsoft Products *See Microsoft Section*		
PANEL Plus *by Roundhill Computer Systems*	795	535
REAL-TOOLS *Binary Version by PCT*	99	89
Library Source Version	599	539
Complete Source Version	999	729
RM/COBOL *by Ryan-McFarland*	1250	949
RM/FORTRAN *by Ryan-McFarland*	750	549
SCO Products *See SCO Section*		

Terms are subject to change.
©1987 Programmers Connection, Inc.

CIRCLE 129 ON READER SERVICE CARD

LOWEST PRICES
Due to printing lead times, some of our current prices may differ from those shown here. Call for latest pricing.

FREE SHIPPING
Orders within the USA (including Alaska & Hawaii) are shipped FREE via UPS. Express shipping is available at the shipping carrier's standard rate with no rush fees or handling charges. To avoid delays when ordering by mail, please call first to determine the exact cost of express shipping.

VISA and MasterCard are accepted at no extra cost. Your card is charged when your order is shipped. Mail orders please include credit card expiration date and authorized signature.

CODs AND POs
CODs and Purchase Orders are accepted at no extra cost. No personal checks are accepted on COD orders. POs with net 30-day terms (with initial minimum order of $100) are available to qualified US accounts only.

SALES TAX
Orders outside of Ohio are not charged state sales tax. Ohio customers please add 6% Ohio tax or provide proof of tax-exemption.

INTERNATIONAL ORDERS
Shipping charges for International and Canadian orders are based on the shipping carrier's standard rate. Since rates vary between carriers, please call or write for the exact cost. International orders (except Canada), please include an additional $10 for export preparation. All payments must be made with US funds drawn on a US bank. Please include your telephone number when ordering by mail. Due to government regulations, we cannot ship to all countries.

VOLUME ORDERS
Volume orders may qualify for additional discounts. Call us for special pricing.

SOUND ADVICE
Our knowledgeable technical staff can answer technical questions, assist in comparing products and send you detailed product information tailored to your needs.

30-DAY GUARANTEE
Most of our products (excluding books) come with a 30-day documentation evaluation period or a 30-day return guarantee. Please note that some manufacturers restrict us from offering guarantees on their products. Call for more information.

MAIL ORDERS
Please include your telephone number on all mail orders. Be sure to specify computer, operating system and any applicable compiler or hardware interface(s). Send mail orders to:

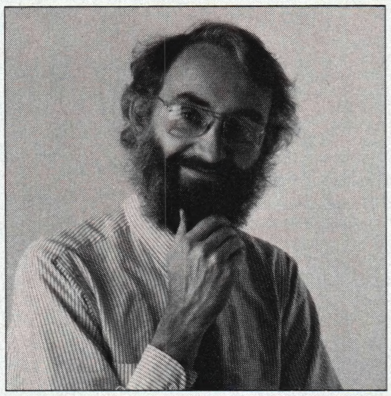
Programmer's Connection
7249 Whipple Ave. N.W.
North Canton, OH 44720

USA	**800-336-1166**
CANADA	**800-225-1166**
OHIO & ALASKA (Collect)	**216-494-3781**
TELEX	**9102406879**
EASYLINK	**62806530**
INTERNATIONAL	**216-494-3781**
CUSTOMER SERVICE	**216-494-8899**

Hours: Weekdays 8:30 AM to 8:00 PM EST.

PROGRAMMER'S QUALITY INTEGRITY Service SUPPORT PRICE CONNECTION

SWAINE'S FLAMES



What do the following have in common: binary addition, subtraction, multiplication, and division; sorting; graph connectivity; multiplication of matrices and finding their inverses, determinants, and ranks; polynomial greatest common denominators; and context-free languages?

Answer: they all belong to a class of problems denoted as NC, short for Nick's class, after one Nicholas Pippenger who gave the class formal definition some eight years ago in a paper called "On Simultaneous Resource Bounds" in the *Proceedings of the 20th IEEE Symposium on the Foundations of Computer Science* (IEEE Computer Society, Los Angeles 1979). The problems that gain admittance to Nick's class are just those problems that can be solved substantially faster and more efficiently using many processors rather than one. They are the problems that will drive the growth and spread of parallel processing technology.

Progress on parallel processing architectures seems to be gaining momentum. Some of the approaches include investigating problems such as the Byzantine Generals problem, in which processors have to reach an agreement through message passing even though some of the messages are unreliable, and research on message passing in a sparse network of processors. And up in the hills behind the Berkeley campus of the University of California, researchers at the Mathematical Sciences Research Institute, the "Camelot of complexity theory," have developed significant new parallel algorithms in recent years.

The Fundamental Algorist, Donald Knuth, tells of a visit to the University of Chicago some years ago. Entering a certain building, he encountered two signs. One said "Information Science" and had an arrow pointing to the right; the other said "Information" and pointed to the left. A real-life cartoon of the schism between science and practice, or, as Knuth sees it, between science and art.

Knuth has argued for years that computer programming should be viewed more as an art and that the toolmakers ought to make tools that are a pleasure to use. Floating-point arithmetic should satisfy simple mathematical laws, he argues; then using it for serious purposes could be pleasant. He even thinks that JCL can be beautiful.

Beauty is in the eye of the beholder. Is reverse Polish notation beautiful? How about those Unix program names? Awk! To some, beauty can be structural elegance, as in Bach's "Tocatta and Fugue in D Minor"; to others, it's a clever hack, like Tom Lehrer's "Poisoning Pigeons in the Park."

Alan Turing, one of programming's patriarchs, loved a clever hack. Turing, as his colleague James Wilkinson put it, "was particularly fond of little programming tricks (some would say he was too fond of them to be a 'good' programmer)." Because the world's first working program on an electronic stored-program computer ran on June 21, 1948, and Turing was writing programs for the machine a few days later, this particular aesthetic disagreement would seem to have deep roots.

All of the above items were drawn from a book that I recommend to any programmer. It's *ACM Turing Awards Lectures: The First Twenty Years: 1966-1985*, (Addison Wesley, 1987). Credit where credit is due.

The key to the puzzle from last month is that the stated conditions imply that only Mickey would get any of the loot. That column was a nostalgic aberration, by the way; I once did puzzles like it weekly, encouraged by Maggie Canon, then editor-in-chief of *InfoWorld*, now of *Macintosh Today*. Thanks, Maggie. Credit where credit is due.

Cousin Corbett's Secrets of Software Success, Part VII: The Rule of Point One. This is a corollary to an earlier Secret, but is important enough to merit a Part number of its own. The rule is: use Version i.0 of your product to debug the changes since Version i-1. When the product is relatively stable again, release it as Version i.1. The user version of this rule is: don't buy anything until Version i.1 is released. The authorship of this rule is in dispute. A *PC Week* author may already have published the rule, but he borrowed it from our editor, Tyler Sperry, who claims to have invented it. My cousin Corbett, on the other hand, swears that it's his. All I know for sure is that the inspiration came from Microsoft. Credit where credit is due.

Michael Swaine

Michael Swaine
editor-in-chief

How A C Programmer Became A Screen Star

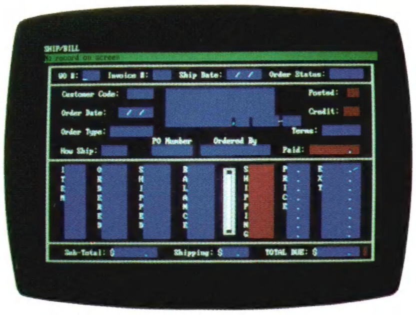
Screens, the Visible Part of Your Program.

A program is often judged by how well the screens are executed. However, the real creativity lies in what goes on behind the screens.

ScreenStar is a product that allows your real creativity to light up the screen. It reduces costly screen, window, and data validation development time.

You Take the Bows, We Write the Code.

Our natural drawing commands allow you to paint any screen imaginable. Press one key when you are satisfied and ScreenStar produces concise, commented, ready-to-compile code. This allows immediate testing of the I/O screens, including smooth, even scrolling between multiple screens.



Create or capture complex screens with data-entry filters built in.

If all ScreenStar did was turn screens into code it would be a useful tool. Yet ScreenStar also permits a wide range of field types. Some of the choices include date, alphanumeric, telephone, yes/no, dollar, time and user-definable fields.

Other valuable data-entry filters are built in, such as required field, display only, and many others. All screen fields are generated with error-checking routines.

ScreenStar Not Only Captures Your Imagination, It Captures Screens.

The memory-resident capture program converts any screen into a ScreenStar file in seconds, including those generated by programs like Dan Bricklin's Demo Program.

ScreenStar Sets the Stage for Windows.

ScreenStar comes with a complete window generating library. You design the help screens and pop-up windows. Essential ScreenStar windowing functions tie them together in one smooth package.

Curtain Call.

They may not ask for your autograph, but they will want to know how you did those screens. Screenstar is more than a screen-painting program. It is a screen processor. No professional programming environment will be complete without this product.

We know you will enjoy using ScreenStar. However, should you give it less than rave reviews, return it within 30 days for a full refund.

★ Interactive screen painting and subsequent code generation.

★ Multiple screen design and scrolling.

★ TSR screen capture program, works with any program including Dan Bricklin's Demo Program.

★ Complete window design including overlapping window functions.

★ Screens are compressed into data structures, and remain a permanent part of the program. No messy data files to look for.

Price - $99

W/Source add $99

Audition Our Product Today. Call:

(201) 762-6965

Essential Software
South Orange Plaza
76 South Orange Ave., Suite 3
South Orange, NJ 07079

CIRCLE 139 ON READER SERVICE CARD

ScreenStar is a trademark of Essential Software Inc.
Dan Bricklin's Demo Program is a trademark of Software Garden Inc.

IBM Spoken Here

and here

and here

and here

CROSSTALK
Mk 4
Serial # 4102196
Version 1.1
System IBM PC & PS/2
© 1987 DIGITAL COMMUNICATIONS ASSOCIATES, INC. All Rights Reserved.

Whatever dialect of IBM you need to speak, CROSSTALK® Mk. 4 makes the connection.

Now, one program does the job that used to require several. CROSSTALK® Mk. 4 allows high-speed direct communications between PCs and minicomputers, or (with an IRMA™ board) between your PC and an IBM Mainframe, or (with Smart Alec™) between your PC and IBM System 3x's. If you like, CROSSTALK can support all of these sessions (and others) simultaneously, and display each session in its own window.

CROSSTALK Mk. 4 emulates all the terminals you're likely to find useful. That includes IBM 3101 (page and character modes), IBM 525x, IBM 529x, IBM 327x, as well as many popular async terminals like the DEC VT100 and VT220 series. CROSSTALK Mk. 4 includes the powerful CASL™ programming language, which allows you to automate communications applications quickly and easily.

So if you're used to thinking of CROSSTALK just to use with a modem, you're missing some important connections. Ask your dealer for details, or write:

dca® Digital Communications Associates, Inc.
1000 Holcomb Woods Parkway / Roswell, Georgia 30076
1-800-241-6393

CROSSTALK COMMUNICATIONS

CIRCLE 171 ON READER SERVICE CARD